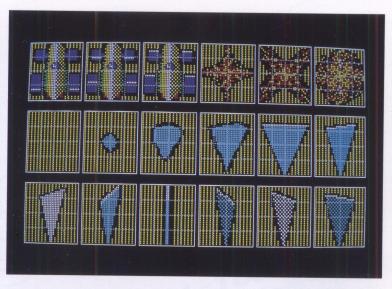


第4章で使用するパターンデータ

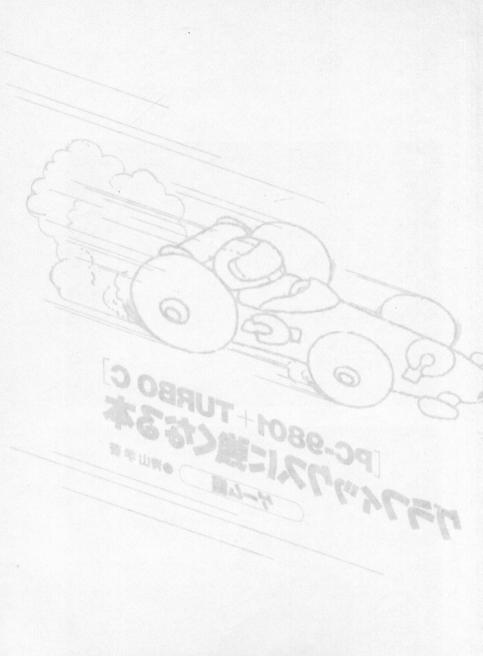
第3章のパターンエディタで作ります。







啓学出版



TURBO C は米国 Borland Intarnational 社の登録商標です MS-DOS は米国 Microsoft 社の登録商標です。 その他本文中に登場する製品名は一般に開発メーカーの商標です。

を できます。 はじめに ハーツ b 付給されている

コンパイラというと、ほんの少し前までは、パーソナルに所有することは夢物語として語られるだけでした。しかしながら、現在では TURBO Cのような統合環境下で、プログラムの編集、実行、デバックができる手軽なコンパイラの登場により、様相は一変してしまいました。しかも、MS-DOS が普及するにつれ、マシンを自在に使いこなしたいという欲求を実現するための有力な手段として、C言語は不動の地位を占めてきたようで

実際、C 言語は様々な欲求に答えてくれます。データベースやエディタだけではなく、コンパラや OS までも C 言語で記述されるに至っては、C 言語のスーパースターぶりを物語っていると言えるのかもしれません。

さて、本書ではグラフィックスの中でも人気の高い分野である TV ゲームをテーマとして掲げています。今までこの分野は、難解なマシン語でしか開発できない特殊な分野とされてきましたが、高速な処理を要求されるリアルタイムゲームでさえも、工夫次第では C 言語で作ることが可能となってきました。もちろんゲームでは、ハードウェアに強く依存したプログラムとなるのは避けられないことですから、対象となる機種は PC-9801シリーズに限ることになります。しかしながら、逆に考えれば、機種に強く依存したプログラムも記述できるというのが C 言語の大きな特徴であるともいえるのです。

本書では、2進数の基礎から始まり、必要とされるハードウェアの知識を解説すると共に、C言語の初心者でも読めるように各関数は簡単なものばかりを用意しました。そして、それぞれの関数は機能別に部品化を図ったものとし、これらをプロジェクトファイルとして組み上げていくことに

よって、お絵かきツール、パターンエディタ、マップエディタ、そして最終的にはリアルタイムゲームを完成するという構成になっています。本書を通して TURBO C の新たなる可能性を開拓する楽しさを、ぜひ味わってください。

なお、本書では、次のようなシステムを想定しています。

- 1 …… PC-9801シリーズ本体
- 2 …… ディスプレイ (640×400ドットのカラーディスプレイ)
- 3 MS-DOS システム
- 4 ······ TURBO C Version 2.0以降

また、PC-9801 VM21以降の機種では640K バイトのメモリを標準で装備していますから問題ないのですが、もしも、640K バイトのメモリが装備されていない機種であれば、サンプルプログラムの中でコンパイルできないものもありますから注意して下さい。

最後になりましたが、本書の執筆に際し、いろいろとご協力くださった、 啓学出版の嶋貫健司氏、パソコンクラブ A.G プロジェクトの大野真也、佐 藤新市、柴田錦見、清水明、寺内一郎の各氏と、処理系を提供していただ いた(株)マイクロソフトウェア アソシエイツの方々に、この場より改めて 御礼申し上げます。

1980-09 は崩勝をなる鬼状。されてすることを1991年7月7日 青山 学

く依存したプログラムも記述できるというのがC書語の大きな特徴であるともいえるのです。

本書では、2進数の基礎から始まり、必要とされるハードウェアの知識を

等級すると共に、C 直轄の初心者でも認めるように各国数は簡単なものは する最高を受けます。 かりを用意しました。そして、それぞれの関数は機能用に部品化を図った。

Bのとし、これら素型は大工文本語のは、Producinaをおれておから立とは

もくじューロ > 110 3 /mmm

はじめに iii 10 メータン もくじ v

第1章 0と1だけの世界 1

2 進数(binary number) 4 2 進数の単位の呼び方 6 文字コード 7 2 進数と16進数 8 マシンの心臓部を探る 10 PC-9801の G.V.RAM とは 11

第2章 実践グラフィックス基礎編 15

グラフィックス処理を始める前に 16 グラフィックス処理を開始するには 18 far 指定のポインタ変数 20 色の混ぜ合わせ 23 ドットから直線へ 25 色の混ぜ合わせ実践編 29 縦方向に直線を描くには 31 ドット座標の定義 35 シフト演算子の効用 39 色の成分チェック 41 キーの入力を知るには 43 カーソルの表示 46 割り込み処理の実際 52 マウスを組み込めば、お絵書きツールのでき上り 53

第3章 パターンエディタに挑戦 61

BOX ルーチンの作成 62 割り込み処理の実際(点滅カーソルの表示) 69 割り込み関数の定義 74 キーの入力を知るには(高速編) 76 G.V.RAM 上のパターンのコピー 95 グラフィック画面のセーブ/ロード 107 ファイル名の入力 113 ディレクトリ内のファイルの探索 115

アニメーション処理 126 キャスト演算子 130 マップエディタに挑戦 133 GDC によるスムーズスクロール 145 ヒーロー登場 155 敵の出現 165 まとめ 171

索引 173

第1章 0と1だけの世界

現在ではコンピュータの技術も IC や LSI、VLSI の時代になりましたが、設計された当初は、外部からのノイズや素子のばらつきなどから信頼性が問題でした。そこでコンピュータは、電流が有るか無いか電圧が正か負かなどの単純な2つの状態を基準として動作するように作られたのです。

たった2つの状態ですが、これだけでも片方を0もう一方を1とすることによって、あらゆる数字を2進数で表現できるわけです。そして、これら2進数の数値に、加算せよとか引き算せよなどの特別な意味を持たせたマシン語コードといわれるもので、命令が組み立てられたのです。

いまではその当時と比べると格段に技術も進歩したのですが、やはり現 在でも多くのコンピュータが同じ仕組みで動いています。

ところで、コンピュータが2進数の数値命令で動いているといっても、人間が直接2進数の数値命令を出しているわけではありません。通常は高級言語と言われる命令でプログラムを記述し、そのプログラムをコンピュータ自身にマシン語コードへと翻訳させているのです。このように、プログラムはどんな言語で書かれたものでも最終的には、マシン語コードに翻訳してから実行されます。なお、マシン語に翻訳する前のプログラムをソースプログラムといいます。

本書で扱っている C 言語も例外ではなく、C 言語で書かれたソースプログラムをなんらかの翻訳者を通して、マシン語へと翻訳したものを実行することになります。この翻訳者の形態は、インタープリタ型とコンパイラ型の 2 つに大別されます。

インタープリタは、ある言語で書かれたソースプログラムをひとつひと つ読み込んでは、その文字列が何をすべきかを解釈して、あらかじめ組ん であったマシン語のプログラムの中から、対応している処理を実行する。 というタイプにつけられた総称です。 この方法では、インタープリタ本体と高級言語で書かれたソースプログラムだけですみますから、メモリは少なくてすみます。ただし、翻訳しながら実行するという過程では、実行時に翻訳時間が加わることになりますから、その分スピードはどうしても遅くなります。

現在のパソコンには、ほとんどインタープリタ型 BASIC が付いてくる ので少なくとも一度はインタープリタ型言語を使ったことがあるのではな いでしょうか。

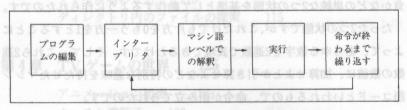


図1-1 インタープリタの実行過程

使ってみるとインタープリタ型言語は非常に手軽に使え、しかも、修正 や変更もいたって簡単におこなえることがわかります。

これに対し、ソースプログラムをあらかじめ全部マシン語に翻訳して、その翻訳したマシン語のプログラムを実行するというタイプをコンパイラ型といいます。コンパイラによってマシン語に翻訳する作業をコンパイルと呼んでいますが、ソースプログラムは、コンパイラによって未定議ラベル等を含む中間的なオブジェクトファイル(拡張子OBJ)に編集されます。さらに、リンカーによって、最終的なマシン語プログラムが作られるのです。この最終的なプログラムを実行プログラム(拡張子EXE)といいます。実行プログラムはインタープリタ型とは違って、実行時に翻訳する時間

プログラ ムの編集 コンパイラ → 実行可能 プログラ ムの編集 プログラ ムの実行

図1-2 コンパイラの実行過程 ロストトトライス

がかかりませんから、たいへん高速に実行されます。

その反面、何らかのバグを生じた場合には、翻訳される前のソースプロ グラムを取り出して修正し、再びコンパイルして実行を確認するという一 連の面倒な作業をしなければなりません。とはいっても、得られる安全性 や高速性は、何物にも代え難いといえるでしょう。

本書ではこのコンパイラ型C言語の特徴をいかしてリアルタイムゲー ムを作りながらC言語の可能性をさぐろうというわけです。

	14	
IIII		

■ 2進数 (binary number)

リアルタイムゲームに限らず、高速なグラフィックス処理をするには、 やはりコンピュータの世界、すなわち0と1だけの世界である2進数(binary number) の基礎的なことを知らなければなりません。2進数とはどのよう なものかを知るために、ここで、4桁の2進数と10進数を対比させてみましょ う。

表1-1 10進数と4桁の2進数との対応

7 - 8 - 1 - 1 - 7 - 9	2.89
0 0000	8 1000
1 0001	9 1001
2 0010	10 1010
3 0011	11 1011
4 0100	12 1100
5 0101	13 1101
6 0110	14 1110
7 0111	15 1111

いかがですか、4桁の2進数で表すことができるのは表1-1にあるように全部で0から15までの16種類です。

このように2進数は、0と1だけが並んでいるだけですから、何がどれを表しているのか人間にとっては区別がしにくいですね。

さらに2進数の世界を知るために、1桁の2進数どうしの足し算を考えてみましょう。

図1-3 1 桁の 2 進数の足し算

はじめの3つの演算は10進数とまったく同じですが、注目したいのは最後の1+1です。これは10進数では2となりますが、2進数の世界ですから2にな

ると桁上がりを起こして10となります。十進数で2と表しても、2進数で10 と表しても物の本質、すなわち、絶対的量にはなんら変わりがないという のは書くまでもありません。

ここで、先ほどの表1-1を再び参照してみましょう。2進数をもっと身近 に感じるために紙とエンピツで2進数の0000から最後の1111になるまで順 に1を足してみてください。

実際に計算をしてみると、なんとなくコンピュータの味け無い世界に触 れたような気がしませんか。

支援さらのに、テトの単位を使っているを対抗。これがいるに目前の2組数はほう



■ 2進数の単位の呼び方

さて、コンピュータ気分に浸ったところで話を先に進めましょう。我々人間の世界では物事の関係を表すのに色々な単位を用いています。たとえば重さを表すのに kg、時間を表すのには時、分、秒、などですが、コンピュータの世界でもいろいろな単位を使っています。

コンピュータの世界の基本単位である2進数1桁のことは、ビット (bit) と呼んでいます。これは binary-digit の略です。取りうる値は0か1の2つだけしかありません。これから本書では1桁の2進数とか2桁の2進数とは言わずに、このビットの単位を使っていきます。したがって1桁の2進数は1ビット、2桁の2進数は2ビットということになります。

2進数4桁ではニブル(nibble)といいます。そして、8桁ではバイト(byte)といいますが、このバイトはコンピュータの世界ではビットと並んでよく使われる単位です。たとえば、PC-9801のメモリ上の記憶単位がバイト単位(8桁の2進数)ですし、マシン語もバイト単位で構成されています。また、英文字や、特殊文字を処理する単位もバイト単位です。

これらのほかにはワード (word) という単位がありますが、これは処理系によって4ビット、12ビット、32ビットなど、いろいろな桁数があります。PC-9801では16ビットをひとつのワードとしています。

これら4つの単位(ビット、ニブル、バイト、ワード)は、コンピュータ の世界では常識として扱われていますから、その言葉の意味を覚えておか なければなりません。

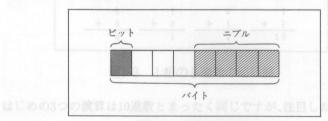


図1-4 2 進数の単位

■ 文字コード

画面上に表示されている英文字や漢字はコンピュータが扱う場合に、やはり2進数として扱っています。ただ、CRTの画面上に表示された時が2進数ではなく、文字の形をしているにすぎません。

すなわちひとつひとつの文字に、2進数を1対1に対応させているのです。 もちろん、2進数1桁では、0か1の2通りしか対応できませんから、ある程度 十分な桁数である8桁の2進数(1バイト)を使っています。

この対応のさせ方はいろいろ考えられますが、処理系やコンピュータに よってばらばらではまったく互換性がなくなりますから、そこにはちゃん と標準化がなされています。

PC-9801では英文字や特殊文字を表すコードとしてアスキーコードに準拠しています。たとえば、英文字 A のアスキーコードは2進数では01000001 となります。 B は01000010、C は01000011です。 ですから、信号として01000001が送られると CRT 上に英文字の A が表示されるわけです。

なお、ソースプログラムはエディタを使って、このアスキーコードで編集します。

5ことが分かります。

1111 + 1 10000 ← 新上がのりが生じる

図1-8 下位 4 ビットに 1 本足す

上のように計算されますが、得られた結果は下位4ピットでは足りずに、 と位4ピットの方へと桁上がりを生じることが分かります。

このことは、2進数の4桁が集まって、ひとつの数字を構成すると見立て

■ 2進数と16進数

ところで、2進数を扱う時には0と1の羅列ですから人間にとってはとても 苦痛です。しかも、書き表す時には、かなりのスペースを使いますので、 むだが多いともいえます。

そこで2進数のおもしろい性質を使って、2進数を16進数で表記するという手品のような、まったく不思議な方法を使うわけです。ここではあっさりとその種あかしをしてみましょう。

通常よく使われるメモリの基本単位である8ビットの2進数を考えてみます。まず代表選手として、00001111という半分が1である状態を考えます。 さらに、この8桁の2進数を上位4桁と下位4桁の2つに分けてみます。

図1-5 8桁の2進数を2つに分ける

この図で、下位の2進数1111は表の1-1より、10進数では15に対応していることが分かります。

さて、この下位の4ビット1111に1を足すとどうなるでしょうか。

図1-6 下位 4 ビットに 1 を足す

上のように計算されますが、得られた結果は下位4ビットでは足りずに、 上位4ビットの方へと桁上がりを生じることが分かります。

このことは、2進数の4桁が集まって、ひとつの数字を構成すると見立て

ることによって、16になれば上位の位へと桁上がりを起こす16進数と同じ 動きをしているとみなせるのです。したがって8ビットの2准数は、2桁の16 進数として表記できるというわけです。

ところで、数字は0から9までしかありませんから、16進表記をする時に は10~15までを、アルファベットの A~F の6文字に対応させています。こ れからは16進数といえばアルファベットの A~F も数字の仲間に加えてあ げてください。

また、16進数と10進数との明確な区別をするために、一般的にはHexadecimal (16進数) の頭文字の Hを16進数の後ろに付けて表記しています。 なお、C 言語で扱う数値は16進数の前に0x を付記することになっていま す。RAMに数値を参加がまりでませるが関連な内状体の内を交叉がある

PC-9801ではモニタを使ってメモリの内容を表示したり、メモリに数値 を書き込んだりすることができますが、この時に威力を発揮するのがこの 16進表記です。また、BASIC の PEEK 文や POKE 文でも16進表記で2進数 を扱うことができます。

このように2進数と16進数は密接な関係にあるのですが、これはあくまで も見かけ上であって、コンピュータは2進数として扱っているのです。

■ マシンの心臓部を探る

マシンの心臓部、すなわち CPU (Central Processing Unit) のことですが、やはり C 言語を十分に利用するには CPU の特徴ぐらいは知っておくべきでしょう。PC-9801シリーズでは、8086系 CPU が使われています。この CPU の特徴は、なんといってもセグメントによるメモリ管理の方法です。

この概念は、メモリを CPU が一度に数えられる範囲で分割し、その分割 単位で管理するという考え方です。この分割単位をセグメントと呼び、そ のセグメント内の相対的な位置をオフセットといいます。

ですからメモリを直接参照する場合には、8086系 CPU 上でプログラムが動いている限り、どのセグメントを使うのかというセグメントアドレスと、そのセグメント内の相対的なオフセットアドレスの2つを与える作業が必要となってきます。PC-9801上で TURBO C をインストールする時に、メモリモデルを指定したと思いますが、このメモリモデルもプログラムでセグメントをどのように使っていくのかの指定に他なりません。

C言語は高級言語であるにもかかわらず、メモリを直接アクセスすることができるようになっています。当然のことながら直接メモリをアクセスする場合には、セグメントアドレスとセグメント内のオフセットアドレスを設定することになります。

さて、本書がテーマとして掲げている TV ゲームは、実は、このメモリ に直接アクセスする作業にほかならないのです。

上のように計算されますが、得られた結果は下位4ビットでは足りす

このことは、2連数の4桁が築まって、ひどつの数字を構成すると見立て

■ PC-9801のG.V.RAM とは

PC-9801のメモリには、グラフィックスを表示するための特別なメモリが用意されています。これをグラフィックビデオラム(以下 G.V.RAM と略記)といいますが、このメモリは、他のメモリと同様にバイト(8ビットの2進数)単位で数値を記憶させることができます。ただ、他のメモリと異なるのは、記憶した数値のビットイメージ(2進数の1桁)が画面上のドット(点)に対応しているということです。

すなわち、CRT画面でグラフィックス処理をするということは、このG. V.RAMに数値を書き込むということなのです。

G.V.RAM は、色別に4つのグループに分かれます。それぞれをプレーンと呼んでいますが、初期状態ではプレーン0は青で、セグメントアドレス $A800_{\rm H}$ 、オフセットアドレス $0_{\rm H}$ から始まっています。一般的にセグメントアドレスとオフセットアドレスは:で区切って表記していますから、この場合には、 $A800:0000_{\rm H}$ となります。また、各プレーンの大きさは $8000_{\rm H}$ バイトとなっています。

同様に、プレーン1が赤で、アドレスはB000:0000 $_{\rm H}$ ~B000:7FFF $_{\rm H}$ 、プレー

	0.20	178	2	***************************************	\$77	78	79
0	0000H	0001 _H	0002 _H		004D _H	004E _H	004F _H
1	0050 _H	0051 _H	0052 _H		009D _H	009E _H	009F _H
	٠.						
	٠.						
	٠.						
			TV画面	(640×400ドット	モード)		
•	٠.						• •
398	7C60 _H	7C61 _H	7C62 _H		7CAD _H	7CAE _H	7CAF _H
399	7CB0 _H	7CB1 _H	7CB2H	***************************************	7CFD _H	7CFE _H	7CFF _H

図1-7 640×400ドットモード表示

 \sim 2が緑で、アドレスは B800:0000 $_{\rm H}\sim$ B800:7FFF $_{\rm H}$ 、そして、プレーン3が輝度用で、アドレスは E000:0000 $_{\rm H}\sim$ E000:7FFF $_{\rm H}$ となります。

ここで、各オフセットアドレスが $0000_{\rm H}$ ~ $7{\rm FFF}_{\rm H}$ となるように、セグメントアドレスを設定してあることに注意してください。

この場合の、各プレーンのオフセットアドレスと TV 画面との対応は次のようになっています。なお、実際に画面に表示されるメモリの範囲は、グラフィックスシステムに 640×400 ドットモードを選択した場合、オフセットアドレスで $0000_{\rm H} \sim 7{\rm CFF_H}$ (図1-7) となります。

	0	1	2	COA Y W. LA. AC ON	77	78	79
) [0000 _H	0001 _H	0002 _H		004D _H	004E _H	004F _H
	0050 _H	0051н	0052 _H	()	009D _H	009E _H	009F _H
43	であれる						10.800
				(640×200ドット	モード)		\$49 *****
7)	67/APC1		TV画面			3Е2Ен	3E2F _H

図1-8 640×200ドットモード第1ページ表示

	0	187 1 ×	2	VXXXXX	77	78	79
0	3E80 _H	3E81 _H	3E82 _H		3ECD _H	3ЕСЕн	3ECF _H
1	3ED0 _H	3ED1 _H	3ED2 _H		3F1D _H	3F1E _H	3F1F _H
	本務が						1
	アクセス						
				(640×200ドット・			
٠		7С61н				7САЕн	

図1-9 640×200ドットモード第2ページ表示

また、 640×200 ドットモードの画面に表示されるメモリの範囲はオフセットアドレスで $0000_{\rm H}$ ~ $3E7F_{\rm H}$ (図1-8)、または、 $3E80_{\rm H}$ ~ $7CFF_{\rm H}$ (図1-9)となります。どちらの範囲を使うかは、ページ指定によって選択します。

プレーンが4つ(青、赤、緑、輝度)ということは単純に考えると、4種類の色しか出せないようにも思えますが、実は、各プレーンの表示アドレスが重なった(同じオフセットアドレスとなる)場合には、別の色を発色する仕組みになっているのです。この重なり方は全部で16種類ありますから、必然的に、色としては16種類が使えることになります。また、これら16種類の色は、固定されているわけではなく、4096色の中から任意の色を設定することができるようにもなっています。

さて、せっかくコンパイラを使っているのに長々と2進数やメモリの話ばかりでは興味も半減かもしれませんね。そろそろ、0と1だけの世界から抜け出すことにしましょう。

1599

等業大(-646×200ドラト起手下の画面に映影像組む大もりの範囲能生メセットアドレスで0606。全3E付着図1989。並影像108度的。人々OFF。(図目9年となります)ともの範囲を使う起収がページ指定にような選択します。

第15十名の金融技術電話を25025年(主意歌、雑食学が25025年) これのベーイズ

図1-8 640×200ドットモード第1ページ表示

38.7Da

	0		2	*************	77	78	79
0	3E80u	32381H	301.824	enter ca grandina antendo entendo en el per-	3ECD _N	SECEN	SEATE!
1	3E.D0 _H	380 lu	3E.O.2n	#86453 - 125 - 127 48 - 1	RF IDe	3F1En	SFIFE
				ji ing kalendari ka		t.	
a.							
a.							
			TVMM	(649×990 Fig.)	先世 [7]		
							*
1							
							.78
198	7C60s	7C618	7C.62n	**********	7CADs	ACAER	7CAFn
199	7CBOs	7CB1st	7CB2n	4-24-24-0-22-2-2-2-2-2	WELL	7CFEs	2CFF 8

図1-9 840×200ドットモード第2ページ表示

第2章 実践グラフィックス基礎編

グラフィックス処理の醍醐味はなんといっても、TV ゲームに代表される2次元+αの世界を創造することでしょう。

新しい世界を創造するのですから、それこそ、神様のような気分が味わえるわけです。しかも、それが製品化されるとなると、もうこたえられません。

この章では、グラフィックスの最も基本となるドット(点)に焦点を当てて話を進めていきます。大海も一滴の水からできているように、あらゆるグラフィックス処理の基本単位はドットですから、このドットを自由に扱うことができれば、グラフィックス処理も自在にこなせることになるのです。

では、TURBO Cを使うための、簡単な下準備から始めることにしましょう。

がの事となりますから、所定のディレクトリに用意し口はGOTROS

なお、本語に掲載されているプログラムだけをプログラミングするならば

これは、PC-9801シリースに、G種類のグサブオやラスジスケムがあるか。 こです。 DC-0801シリーズのグラフィックフシステムは、DC-08011/11V/

VF/VM b6, 7974974-94-(GRCG)b6, VX b6 Ux > vx

プトグラフィックチャージャー (EGC) が登載されました。

ですから、、ユーザーのシステムに EGC が登載されていれば PC98EGC.

BGI を、GRCG が登載されていれば PC98GRCG、BGI を、そして、GRCG

+ + 0 -0 -12

なお、グラフィックスシステムが何かを判定する関数 detectgraph()が

■ グラフィックス処理を始める前に

TURBO C に限らず、他の MS-DOS 上で動いているシステムにとって、グラフィックス処理を始める場合には、必ずグラフィックスシステムの初期化作業をしなければなりません。

というのも、MS-DOSでは、グラフィックスシステムもひとつのデバイスとみなしているため、BASICのように初めからグラフィックスシステムが使える状態ではないのです。

一般的には、MS-DOS 付属のグラフィックスドライバをデバイスドライバに組み込んで制御することになります。

これに対し、TURBO C では、MS-DOS のグラフィックスドライバとは 無関係に、独自の BGI(Boaland Graphics Interface)というライブラリ を用意しています。これらのドライバには、次の3種類があります。

PC98.BGI

PC98GRCG.BGI

PC98EGC.BGI

これは、PC-9801シリーズに、3種類のグラフィックスシステムがあるからです。PC-9801シリーズのグラフィックスシステムは、PC-9801U/UV/VF/VMから、グラフィックチャージャー (GRCG)が、VXからはエンハンストグラフィックチャージャー (EGC) が登載されました。

ですから、ユーザーのシステムに EGC が登載されていれば PC98EGC. BGI を、GRCG が登載されていれば PC98GRCG. BGI を、そして、GRCG も EGC もなければ PC98. BGI というグラフィックスドライバを使うことになります。

なお、グラフィックスシステムが何かを判定する関数 detectgraph()が 用意されていますから、特にユーザーがどのシステムを使っているかを意 識することなしに、これらのドライバを使うこともできます。

ところで、これらの*.BGIというファイルは、そのままの形ではなく、 マスターディスクの DISK 3にある BGI.ARC 内に圧縮して格納されてい ますから、使用する場合には圧縮前のファイルに戻さなければなりません。 圧縮前のファイルに戻すといっても、難しいことではなく、同じディスク にある UNPACK. EXE を使って簡単に復元することができます。

次にAドライブからBドライブへのファイルの展開例を示しておきま すから参考にしてください。

まず、DISK 3をドライブ A に、ワークディスクをドライブ B にセット し、次のように実行します。

BGI.ARCの展開例

A>UNPACK BGI.ARC B:*.BGI

展開した*.BGIのファイルは、以下のプログラムではカレントディレ クトリにあるものとして扱っていきます。

グラフィックス関係の関数を使うためには、この他に、TURBOC付属 のライブラリファイル GRAPHICS.LIB と、ヘッダファイル GRAPHICS.H が必要となりますから、所定のディレクトリに用意しておいてください。 なお、本書に掲載されているプログラムだけをプログラミングするならば、 メモリモデルはスモールモデルとなります。

■ グラフィックス処理を開始するには

さて、これでグラフィックス処理の準備がやっと整いました。さっそく TURBOCの関数を使って、グラフィックスシステムの初期化作業をする 関数を作ってみましょう。

```
LIST2-0 C
      グラフィックスシステムの初期化
#include <graphics.h>
                       /* 必要とされるヘッダファイルをincludeする */
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
                              /* グラフィックスモード格納用 */
int gdriver,
gmode: /* グラフィックスドライバ格納用 */
void grphinit (void)
   int ercode;
                              /* 使用変数の宣言 */
   detectgraph(&gdriver, &gmode); /* ドライバ&モードの設定 */
   initgraph (&gdriver, &gmode, "");
                              /* グラフィックスシステムの初期化 */
                              /* エラーコードの取得 */
   ercode = graphresult();
if(ercode!= grOk) { /* エラーであればメッセージを出す */
      printf("¥nグラフィックスエラー:
                             %s¥n", grapherrormsg(ercode));
      printf("Press any key\n");
      getch();
                              /* キーの入力待ち */
      exit(1);
                              /* MS-DOS^ */
```

このプログラムの、実行文の始まりにある関数 detectgraph()で、最適なグラフィックスドライバとモードを決定しています。ドライバとモードを格納する変数は他の関数でも参照することが予想されますから、関数の外で定義します。

次の initgraph()という関数が、ここでメインとなるグラフィックスシス

テムを初期化している関数です。関数 initgraph()の3番目の引数"は、先に出てきたグラフィックスドライバの*.BGI の格納されているディレクトリパスを示しているのですが、この例のように、なにも指定しない場合にはカレントディレクトリに、グラフィックスドライバが入っていることを意味しています。

initgraph()の実行結果は、関数 graphresult()で得られますから、その結果にしたがって、エラー処理をします。

ここでは、単純にメッセージを表示して関数 exit (1)で、呼び出された親プロセスへ制御を戻しています。もし、エラーメッセージでプログラムが終了した場合には、もう一度、各ファイルの存在を確かめてください。

ひとつの画面が構成されるのです。また、PC-9801を引後感ではなるの題間

「A 一回間で第一回回の一起(CLysulging)を 1000では第一回回のみ)用 さされており、ボート A4m、A6mでコントロールしています。

表5-5 画面の // ロコントロール

#include sdos.h> nOA /* 公果とされる動かで本々をそれた自動代出 動代出 動代出 動代出 がは fax *ならいで、面画一帯 示表面画生業・国用ポインをの業質 */ char製品開ポインタの業質 */ that fax *i red; /* int 数は簡単ポインタの業質 */

それぞれの多差。父の大馬をは2000人パイトなのですが、面前上に表示さ

色モードとなり、4プレーン全部使うと16色モードとなりますが101 5100

各メモリは8ビット単位で構成され、そのビットイメージがTV 商面

■ far 指定のポインタ変数

プレーン	開始セグメン トアドレス	開始オフセッ トアドレス	VRAM の大きさ	対応色
第0プレーン	A800 _H	0000н	8000 _H	青色
第1プレーン	B000 _H	0000н	8000 _H	赤色
第2プレーン	В800н	0000н	8000 _H	緑色
第3プレーン	E000 _H	0000н	8000 _H	輝度

表2-1 G.V.RAM アドレスとプレーンとの対応

表2-1に示したように、G.V.RAM にはプレーンが4つ(初期の PC-9801 では青色、赤色、緑色の3つ) あります。これらの4つのプレーンにより、ひとつの画面が構成されるのです。また、PC-9801シリーズでは、この画面が第一画面と第二画面の二組(PC9801、PC-9801U では第一画面のみ)用意されており、ポート $A4_H$, $A6_H$ でコントロールしています。

ポート値出力値	А4 _н	А6 _н
0	第一画面表示	第一画面アクセス
1	第二画面表示	第二画面アクセス

表2-2 画面の I/O コントロール

それぞれのプレーンの大きさは $8000_{\rm H}$ バイトなのですが、画面上に表示される範囲は、 640×400 モードでは、 $0\sim7{\rm CFF_H}$ となり、 640×200 モードではその半分となります。また、第 $0\sim$ 第2プレーンの3プレーンだけを使うと、8色モードとなり、4プレーン全部使うと16色モードとなります。

各メモリは8ビット単位で構成され、そのビットイメージが TV 画面上

のドット(点)に対応しています。

以上が、1章でのグラフィックビデオラム (G.V.RAM) に関しての概要です。なお、これら4つのプレーンは、以後、初期状態の色別に呼ぶことにします。ですから、プレーン0は、Blue(青)の頭文字をとって B 面、プレーン1は、Red (赤) の R をとって R 面、プレーン2は Green (緑) の G 面、そして、プレーン3は Intensity (輝度)の I 面と呼ぶことにします。このほうが、 $0\sim3$ と呼ぶよりも感覚的に色の構成を把握しやすいからです。

さて、C言語でこれらG.V.RAM ヘアクセスするためには各プレーン にたいするポインタを介してアクセスすることになります。

ポインタとは字の通りに何かを指している変数です。変数を指していたり関数を指していたり色々ですが、ここではG.V.RAMというメモリを指していることになります。しかも、セグメントアドレスとオフセットアドレスを設定しなければなりませんから、ポインタは必然的に far タイプのポインタとなります。

次の LIST2-1が、各 G.V.RAM に対する far タイプのポインタ変数を 設定するための関数です。

```
LIST2-1.C
 char型とint型のG.V.RAMへのfarポインタの作成
                       /* 必要とされるヘッダファイルをincludeする
#include <dos.h>
                         /* char型B面用ポインタの宣言 */
char far *c blue;
                         /* int 型B面用ポインタの宣言 */
int far *i blue;
                         /* char型R面用ポインタの宣言 */
char far *c red;
                         /* int 型R面用ポインタの宣言 */
int far *i red;
                         /* char型G面用ポインタの宣言 */
char far *c green;
                         /* int 型G面用ポインタの宣言 */
   far *i green;
                  /* char型I面用ポインタの宣言 */
char far *c_itsty;
                         /* int 型I面用ポインタの宣言 */
int far *i itsty;
void pointer set (void)
```

```
i blue
                               /* char型B面用ポインタの作成 */
        = MK FP (0xa800, 0);
 c blue = MK FP(0xa800, 0);
                               /* int 型B面用ポインタの作成 */
 i red
        = MK FP (0xb000, 0);
                               /* char型R面用ポインタの作成 */
                               /* int 型R面用ポインタの作成 */
 c red = MK FP(0xb000, 0):
 i \text{ green} = MK FP(0xb800, 0);
                               /* char型G面用ポインタの作成 */
 c 	ext{ green} = MK FP(0xb800, 0);
                               /* int 型G面用ポインタの作成 */
i itsty = MK FP(0xe000, 0);
                               /* char型I面用ポインタの作成 */
 c itsty = MK FP(0xe000, 0);
                               /* int 型I面用ポインタの作成 */
```

ここで、far タイプのポインタ変数を定義した所が、関数の外部であることに注意してください。定義したポインタは、色々な関数で使いますから、変数の可視性を考慮して、関数の外部で定義します。すなわち、他の関数で参照されることを前提としているわけです。

far タイプのポインタ変数に対するアドレスは、TURBO Cの関数 MK _FP()を使って設定します。また、ひとつのプレーンに対して int 型と char 型の二種類の変数を用意しています。

このように、ひとつのプレーンに対してポインタを二種類にしたのは、メモリが1バイト(8ビット)単位で数値を記憶することに起因します。すなわち、メモリに対する最小のアクセス単位は8ビット長の char 型となるため、char 型は最も細かいグラフィックス処理をするのに適しているわけです。そして、int 型はこの場合16ビット長ですから、メモリの記憶単位を一度に2つ処理するためにグラフィックス処理の高速化が図れるというメリットがあるのです。

これらの G.V.RAM へのポインタは、いわば、油絵における絵筆のようなものです。絵筆は色別にすれば、青、赤、緑、輝度、の4種類となります。それぞれに、対応したポインタを、blue, red, green, そして intensity は長すぎますから itsty と名付け、int 型の i と char 型の c を頭に付けてあります。

さて、これらの4つの色は混ぜることもできます。といっても、絵の具のようにパレット上で混ぜ合わせるわけではありません。混ぜようとする各ポインタの同じオフセットアドレスに、データを書き込むことにより実現します。

たとえば、赤と青のプレーンの同じオフセットアドレスに書き込めば、 紫色が発色するのです。これら色の混ぜ合わせ方は、表2-3のように16通り あります。

この表2-3のように、4つのプレーンの重なり方は16通りありますが、ここで PC-9801シリーズでは4096色が使えることを思いだしてください。実は、この16種類の色は仮りの色だったのです。ですから、これら16種類のプレーンの重なり方に対して、この表の通りに色を割り付けてあっただけのことなのです。しかし、まったく色がばらばらでは、話の進めようがありませんから、本書ではこの初期状態の色を基準に話を進めていきます。

=	=	0		Γ
- 7	長	~	-	
1	-/	ᆫ		L

青	赤	緑	輝度	色。	青	赤	緑	輝度	色
×	×	×	×	黒	×	×	×	0	薄い黒
0	×	×	×	青	0	×	×	0	薄い青
×	0	×	X	赤	×	0	×	0	薄い赤
0	0	×	×	紫	0	0	×	0	薄い紫
X	×	0	×	緑	×	×	0	0	明い緑
0	×	0	×	水色	0	×	0	0	明い水色
X	0	0	×	黄色	×	0	0	0	明い黄色
0	0	0	×	グレー	0	0	0	0	明い白

(注) 8色モードの時は、青、赤、緑の3プレーンが使われます。

さて、これら16通りの色、すなわち絵筆の色を把握したら、次は筆の動かし方です。では次の意味を考えてみてください。

c_blue + 1; /* 8ビット長のchar型のfarポインタ変数を1増やす */ i_blue + 1; /* 16ビット長のchar型のfarポインタ変数を1増やす */

ここで、 i_blue , c_blue は共に、LIST2-1で設定した B 面に対する far 指定のポインタ変数です。B 面は初期状態では青に対応しています。両者 共に、B 面を指しているポインタです。

それぞれは、まったく同じように見えますが、これらの違いは何でしょうか? それはポインタの指し示しているフィールドの扱い方です。すなわち、プレーン0の G.V.RAM を8ビット長の char 型で取り扱うのか16ビット長の int 型で扱うのかという違いなのです。この違いだけはしっかりと把握しないと混乱をまねきますので注意してください。

ここに示した式は、式としてはどちらも変数を単純に+1するという意味ですが、この場合の変数はポインタ変数ですから式の意味するところが変わってきます。ポインタ変数の場合、1増えることは、フィールド内の次のデータを指すことになるのです。

ですから、フィールドを8ビット長の char 型とすれば、次の8ビットのデータを指し、16ビット長の int 型であれば、次の16ビット長のデータを指していることになります。もし、関数のエントリーを指しているポインタであれば、次の関数を指すことになるのです。

プログラムでは見かけ上同じに見えても、結果的に、メモリの刻み方は 16ビット長のほうが、8ビット長の二倍の刻みとなります。

■ ドットから直線へ

言葉だけではわかりにくいので、実際にプログラムを組みながら、これ らのこと(筆の動かし方)を考えてみましょう。

まず、グラフィックス処理の記念すべき初めの一歩として、画面の一番 上に、横方向の直線を描くプログラムを作ってみました。メイン関数とプロジェクトファイルは次のようになります。

```
LIST2-2.C
 char型のポインタを使って横に青い直線を描く
図1-7から、メモリの並びは画面上橋方向に対応し、画面のひとつの 1*
#include <graphics.h> /* 必要とされるヘッダファイルをincludeする */
#include <comio.h>
extern void grfinit (void); /* 外部関数の宣言 */
extern void pointer set (void);
                  /* char型のB面用ポインタの宣言 */
extern char far *c_blue;
void main(void)
             /* 使用する変数の宣言 */
  int i;
                  /* グラフィックスシステムを初期化する */
 grphinit();
                   /* ポインタをセットする */
  pointer set();
                   /* 1ライン分の0xffを描き込む */
  for(i = 0; i < 80; ++i) {
                  /* B面へ0xffを描き込む */
    *(c blue + i) = 0xff;
  getch(); /* キーの入力待ち */
                   /* グラフィックスシステムの終了とする
  closegraph();
```

```
LIST2-2.PRJ
```

```
LIST2-0
LIST2-1
LIST2-2
```

ここでは、プロジェクトファイル名を、LIST2-2.PRJ としましたが、これは適当に設定してください。プロジェクトファイルを TURBO C へ登録 (GRPH+P) したら、コンパイル&実行です。以後、プロジェクトファイル名は、main 関数のファイル名に、"PRJ" という拡張子を付けたものとします。なお、プロジェクトファイルはメイン関数とは別のファイルとしてディスク上にあらかじめ保存しておきます。

実行してみると CRT 画面の上方に青い横線が描けたはずです。さて、画面上の点をドットいいますが、このドットは G.V.RAM のメモリのビットイメージに対応しているというのは、もう知っていますね。ドット(点)の集まりは1本の線になるのは書くまでもありませんが、ここでメモリと画面との対応をもう一度、図1-7で確認してください。

図1-7から、メモリの並びは画面上横方向に対応し、画面のひとつのラインはメモリ総数0x50バイト(十進数で80)で構成されることがわかります。また、0x50バイトを超えた時には、メモリは連続していますが、画面上は不連続となり次のラインの左側へ移っています。このことは、画面上 Y 方向にメモリを刻んだ場合、0x50バイトのメモリ増加になることを示しています。

以上のことを踏まえ、LIST2-2を参照してみると、どのようなプロセスで青い線を描いたのかがよくわかると思います。

ところで、LIST2-2ではポインタ変数を直接動かす(変数の数値を増やす)のではなく、i という変数を作用させて間接的に動かしています。こうすれば、各ポインタ変数の値は常に一定値となり、TV 画面左上に固定されることになるからです。いわば、版画における見当がつくということなのです。

さて、実際に G.V.RAM メモリに描き込んでいる命令は*(c_blue + i) = 0xff です。LIST2-1で定義した青色の絵筆に相当するポインタ変数は、i_blue または、c_blue でした。どちらを使っても青い線を描くことはできます。ここで問題となるのが、これらポインタの扱い方です。LIST2-2では c_blue を使いましたから、例として i_blue を使ったもの

を次に示します。なお、16進数1桁は1章でもやりましたが4ビットに相当しています。すなわち、16進数1桁で画面上の4ドット(点4つ分)に対応しているわけです。

```
LIST2-3.C
  int型のポインタを使って横に青い直線を描く
*/
#include <graphics.h> /* 必要とされるヘッダファイルをincludeする */
#include <comio.h>
extern void grfinit(void); /* 外部関数の宣言 */
extern void pointer set (void);
extern int far *i blue;
                          /* int型のB面用ポインタの宣言 */
void main (void)
                          /* 使用する変数の宣言 */
 int i,j;
  grphinit();
                          /* グラフィックスシステムを初期化する */
                          /* ポインタをセットする */
  pointer set();
   for(i = 0; i < 40; ++i) { /* 1ライン分0xffffを描き込む */
                          /* B面へ0xffffを描き込む */
      *(i blue + i) = 0xffff;
                          /* キーの入力待ち */
   getch();
                           /* グラフィックスシステムの終了とする */
   closegraph();
```

LIST2-3.PRJ

LIST2-0 LIST2-1 LIST2-3

この例では、ポインタに16ビット長の int 型を使い筆を動かすループ回数は LIST2-2の半分の40とし、さらに、ポインタの指しているフィールドに格納する数値は、16ビット長の数値で、2進数で表すとすべてが1となる0xffff(十進数で65535)としています。

このように、ポインタ変数を換えた分、ループ回数が半分となりました

から、約半分のスピードで処理したことになります。LIST2-2、LIST2-3とを比較すると、これらポインタ変数の c _ blue、i _ blue という絵筆の動かし方の違いがよく分かるのではないでしょうか。

LIST2-3.PRJ

■ 色の混ぜ合わせ実践編

次に、色の混ぜ合わせを、プログラムを通して実際に試してみます。 LIST2-4です。

```
LIST2-4.C
色の混ぜ合わせ
>うごとです。すなわち各ポインタに加策している数値が同じなので√*。
#include <graphics.h> /* 必要とされるヘッダファイルをincludeする */
#include <conio.h>
extern void grfinit(void); /* 外部関数の宣言 */
extern void pointer set (void);
extern int far *i blue; /* int型B面用ポインタの宣言 */
extern int far *i_red; /* int型R面用ポインタの宣言 */
                       /* int型G面用ポインタの宣言 */
extern int far *i green;
void main(void)
int i; /* 使用する変数の宣言 */
  grphinit();
                      /* グラフィックスシステムを初期化する */
                       /* 各ポインタを初期化する */
  pointer set();
  for(i = 0; i < 40; ++i) { /* 1ライン分0xffffを描き込む */
     *(i_blue + i) = 0xffff;
                      /* B面へ0xffffを描き込む */
     *(i red + i) = 0xffff;
                       /* R面へOxffffを描き込む */
  73名で、ヌキリアカシスから
                       /* キーの入力待ち */
  getch();
  closegraph();
                      /* グラフィックスシステムの終了とする */
```

```
LIST2-4.PRJ
```

```
LIST2-0
LIST2-1
LIST2-4
```

宣言部を除けば、プログラムとしては、LIST2-3にR面をアクセスするための一行を加えただけです。

このプログラムが実行することは、最初に青い線を描き、青い線と画面 上同じ位置(同じオフセットアドレス)に、赤い線を描くという作業です。

実際に、このプログラムを実行してみると画面上に表れた直線は紫色となっているはずです。これは、画面上同じ位置に、青、赤、を発色したため、青と赤を混ぜ合わせたことになって紫色の直線となったわけです。ここでのポイントは各色別の絵筆が、CRT画面上、同じ位置に線を描いたということです。すなわち各ポインタに加算している数値が同じなのです。

このように色を混ぜ合わせるプロセスは、CRT画面上同じ位置の必要なプレーンに対して、数値を書き込むことによって簡単に実現できるのです。なお、この各プレーンの組み合せ方による色合いは表2-3を参照してください。

参考のために、LIST2-4.Cに輝度用のポインタを使った場合や、その他、 色々な組み合せで調べみるのもよい方法だと思います。

(blov) niam blov

grphinit(); /* グラフィックスシステムを初頭化する。 pointer set(): /* 本がインタを知識化する。/

for (1 = 0; 1 < 40; ++1) (/* 1ライン分のxffffを描き込む */

* (1_blue + 1) = 0xffff; /* 8間へ0xfffを指さ込む */

/* UKe描音2220/图8 */ :32220 = (1 + ber_1)*

etch(); /* +-の入力待ち */

3losegraph(); /* ゲラフィックスシステムの終了とする

LISTS & PRJ

LIST2-0 LIST2-1

■ 縦方向に直線を描くには

横方向に直線を描きましたから、次に縦方向の直線を描くことを考えてみましょう。また、この時の CRT 画面上の直線の位置としては、一番左側とします。

まず、方針を立ててみようではありませんか。図1-7を参照すると、画面上一番左の一番上に対応するオフセットアドレスは0であることがわかります。

ですから、G.V.RAM上のオフセットアドレスの初期値は0から始めればよさそうです。では、その次の連続したドットはどこのオフセットアドレスに書き込めばよいのでしょうか。やはり図1-7より、メモリ上0x50刻んだ所が縦方向に進んだ次に位置するメモリですから、オフセットアドレスは0x50であることがわかります。同様に縦方向に線を描く場合のアドレスは、

0, 0x50, 0xA0, 0xF0,

となります。したがって、アドレス変化はG.V.RAMの初項0に単純に0x50を加算していくプログラムとなりそうです。

これでメモリの刻み方の方針がたったようなので。次に、このG.V. RAM へどのような数値を格納するかが問題となります。

ところで、メモリアクセスの最小単位は8ビット長の char 型です。すなわちひとつのメモリで8ドット分まとめて記憶するのです。横方向では、これでもまったく問題がなかったのですが縦方向の場合簡単ではなさそうです。

実は、グラフィックス処理をする時にはメモリのアクセス単位が8ビット 長であるということがメリットでもあり、また、最大の問題点ともなって くるのです。ここで LIST2-5.C を参照してください。

LIST2-5.Cでは、ポインタにchar型を使っています。最後までの方針

```
LIST2-5.C
      縦に青い直線を描く
                        /* 必要とされるヘッダファイルをincludeする
#include <graphics.h>
#include <comio.h>
extern void grfinit (void);
extern void pointer set (void);
extern char far *c blue; /* char型のB面用ポインタの宣言 */
void main (void)
(公路) 公司到额
                            /* 使用する変数の宣言 */
   int i, j;
                            /* グラフィックスシステムを初期化する
   grphinit();
                           /* ポインタをセットする */
   pointer set();
                            /* 縦方向へ0x80を描き込む */
   for (i = 0, j = 0; i < 399; ++i, j += 80) {
                            /* B面へ0x80を描き込む */
   *(c blue + j) |= 0x80;
                            /* キーの入力待ち */
   getch();
                            /* グラフィックスシステムの終了とする */
   closegraph();
```

```
LIST2-5.PRJ
```

LIST2-0 LIST2-1 LIST2-5

をたてる前にプログラムを示しましたが、ジックリとプログラムを読んで みてください。

ここで、ひとつのG.V.RAMメモリに格納されている8ビットの画面上の対応を調べてみましょう。

図2-1で示したようにひとつのメモリ上8ビットのデータの内、最上位のビットは画面上では左側に位置しています。

LIST2-5.Cでは一番左側の線を描くことを目的としていますから、 CRT上一番左側に位置するメモリに格納されている数値の最上位ビット だけが1であればよいわけです。2進表記すると次のようになります。

1 0 0 0 0 0 0 0

これを16進表記したものは0x80となります。したがって、この数値をメモリに書き込めばよいのです。

しかし、さらに注意が必要なのです。メモリに0x80をそのまま書き込んだ場合、目的のビット以外は0ですから、1ドットだけを書き込みたいのに、他の無関係な7ドット分も0、すなわち、同じメモリにある他のビット情報を壊してしまうことになるのです。

ここでひとつの工夫が必要になってきます。これには論理演算という巧妙なテクニックを使うのです。もし、LIST2-5.Cのプログラムを読んでこのテクニックに気が付いた方は、もう本書を卒業してもよいくらいのテクニックなのです。

サンプルプログラムの中で G.V.RAM へ数値を書き込んでいる部分に 着目してください。ここでは、この問題点を解決するために、メモリに対 して OR をとっていたのです。こうすればメモリ上の目的のビットだけを 1に、すなわちドットを描くことができるのです。

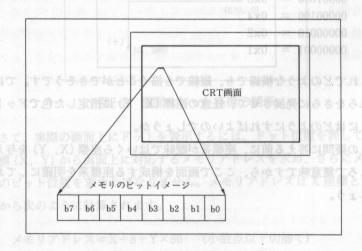


図2-1 メモリのビットイメージ

メモリとの演算に OR を利用する

縦線を描く場合には、この作業を画面上、縦方向に展開することになります。G.V.RAMへのポインタのメモリ上 Y方向増分は、初めの方針の通り、0x50ですから、メモリを0x50毎に刻んで、メモリに格納すべき数値である0x80との OR をとれば、目的の縦線が描けることになります。

これらを実現したのが LIST2-5.C だったのです。では、右方向に1ドットずらした縦の直線を描く場合にはどうなるのでしょうか。これは、ビットが右にひとつずれたデータである0x40を描き込むことになります。さらに右にずらすと0x20となります。参考のために、ひとつのビットだけが1となる8桁の2進数と、16進数の対応を次に示します。

 $\begin{array}{rcl}
 10000000 &=& 0x80 \\
 01000000 &=& 0x40 \\
 00100000 &=& 0x20 \\
 00010000 &=& 0x10 \\
 00001000 &=& 0x8 \\
 00000100 &=& 0x4 \\
 00000010 &=& 0x2 \\
 00000001 &=& 0x1
 \end{array}$

これでどのような横線でも、縦線でも描くことができそうです。では、 これらをさらに発展させて、任意の座標(X, Y) に指定した色でドットを 描くにはどのようにすればよいのでしょうか。

この疑問に答える前に、座標系が曖昧ではいくら座標(X, Y)を与えた ところで無意味ですから、ここで画面を構成する座標系を明確にしておき ましょう。

■ ドット座標の定義

今、画面横方向をX軸、鉛直方向にY軸を考えてみてください。この場合、メモリの構造から画面左上がこの座標系の原点となります。また、X軸の+方向は画面に向かって右方向となり、Y軸の+方向は画面下方向とするのが一般的となります。

ところで、メモリの記憶単位が1バイト (8ビット) であることは、もう知っていますね。また、横の1ラインは0x50 (十進数では80) バイトで構成されていることも知っています。ですから、X 軸の刻みをドット単位、すなわち、G.V.RAM 上のビット単位で考えると、X 軸方向のドット総数は8×80=640となります。

また画面上のライン数の最大値は400ですから Y 軸方向のドット総数は $400(640 \times 200$ ドットモード選択時は200)となります。

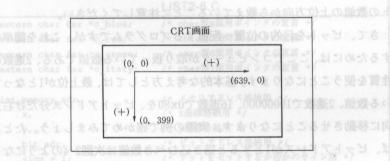


図2-2 画面とドット座標の関係

さて、実際の画面上にドットを表示するには、ドット位置を表している 座標(X, Y)から画面上に対応するメモリアドレスを求め、さらにメモリ 内のビット位置を求めねばなりません。メモリアドレスは X 座標と Y 座 標から次のように計算されます。

メモリアドレス=X:8+Y×80 (小数点以下切捨て)

ただ、これだけではバイト単位のメモリアドレスが求まるだけです。まだ、メモリの1記憶単位である8ビット上の、どの位置にビットを配置するかというビットアドレスを求める問題が残っています。

このビットの位置は、先ほどのドットのX座標を8で割った余りとして 求めることができます。次の図を参照してください。

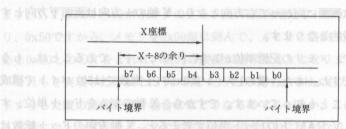


図2-3 メモリとビットアドレスの対応

ビットアドレスの取りうる値の範囲は8で割った余りですから、必然的に0~7となります。この時、ビットアドレスはメモリの記憶単位である8ビットの数値の上位方向から数えていることに注意してください。

さて、ビットを目的の位置へ配置するプログラムですが、これを簡単にするためには、ここでコンピュータが取り扱っている数値である、2進数の性質を使うことになります。基本的な考え方としては、最上位が1となっている数値、2進数で10000000、16進数で0x80を、ビットアドレス分だけ右方向に移動させることになります。実際の例で確かめてみましょう。たとえば、ビットアドレスが1とすると描き込むべき数値は次図2-4のようになります。

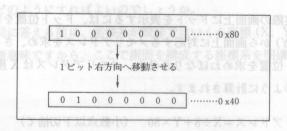


図2-4 ビットアドレスが 1 の場合の例

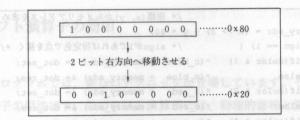


図2-5 ビットアドレスが2の場合の例

さらに、ビットアドレスが2の場合には図2-5のようになります。

ですから、画面上にドットを表示するには、X 座標を8で割った余りとして求められるビットアドレス分、0x80を右方向にシフトして求められる数値と、あらかじめ求めておいた G.V.RAM 上のメモリアドレスとの間で、論理演算 OR を実行するということになるのです。これらをプログラムにしたのが次の LIST2-6です。

```
LIST2-6.C
                     /* char型B面用ポインタの宣言 */
extern char far *c blue;
                    /* char型R面用ポインタの宣言 */
extern char far *c red;
                     /* char型G面用ポインタの宣言 */
extern char far *c green;
                     /* char型I面用ポインタの宣言 */
extern char far *c_itsty;
                     /* メモリアドレス格納用 */
int memory_adr,
  /* y座標格納用 */
                     /* ビットアドレス格納用 */
  bit adr,
                       ドットをセットするか否かのサイン用 */
  sign = 1;
void dot plot (int color)
            になるのです。なお、この複雑は、
int i, j; /* 使用変数の宣言 */
  unsigned char
   dot_set = 0x80,
dot_res = 0;
                     /* 変数bit adrへx÷8の余りを求める */
   bit adr = x \& 7;
                   /* 0x80をビットアドレス分右方向へシフトする */
  dot set >>= bit adr;
                     /* リセット用データの作成 */
   dot res
        ^= dot set;
```

```
/* 座標(x, y)からメモリアドレスを求める */
   memory adr = (x >> 3) + y * 80;
   if(sign == 1) {
                        /* signが1であれば指定色で点を描く */
if(color & 1) *(c blue + memory_adr) |= dot_set;
      else
                    *(c blue + memory adr) &= dot res;
                    *(c red + memory adr) |= dot set;
      if (color & 2)
else
                  *(c red + memory adr) &= dot res;
      if (color & 4)
                    *(c green + memory adr) |= dot set;
      else *(c green + memory adr) &= dot res;
      if(color & 8)
                    *(c itsty + memory adr) |= dot set;
                    *(c itsty + memory adr) &= dot res;
15 () 命位上院的82
```

■シフト演算子の効用

このプログラムでは、新しいテクニックが登場しています。それは、シフト演算子の用法です。このシフト演算子には、物理的意味と算術的な意味があります。

物理的意味としては、単純に数値の桁をずらすということになります。 そして、この場合の数値はコンピュータにとっては2進数のことですから、 2進数の数値の桁をずらすことになるのです。

もう説明するまでもありませんが、メモリに格納する数値の初期値を 0x80 (2進数では10000000となる)とし、X座標を8で割った剰余で右方向 にシフトさせれば、正しいドット位置が設定できるというわけです。

なお、8で割った剰余は2進数の性質から、7との AND により求めることができます。ちょっとしたテクニックとして覚えておきましょう。

シフト演算子には、もうひとつ算術的な意味があります。ご存じのように数値の桁をシフトするということは、右方向にシフトさせればその数値の基数で割ることを意味し、左方向にシフトすれば基数倍することになるのです。

2進数の基数は2ですから、右方向に一回シフトさせれば2で割ることを意味し、左方向に一回シフトさせれば2倍になるというわけです。

この性質を使って $(X \div 8)$ を実現しているのが、LIST2-6.C.の $(X \Rightarrow 3)$ です。これは3回右方向にシフトするという意味ですから結果的に $(X \div 8)$ が求められることになるのです。なお、この演算は、単純に割り算するよりもはるかに高速になります。

ところで、このシフト演算子を使う場合には、注意しなければならない 部分があります。まず、LIST2-6.Cで、初期値0x80を格納している変数 dot _set の宣言部分に着目してください。

(unsinged char)となっています。これは、この変数の数値の符号は考慮しないことを宣言しているのです。実は、符号を考慮すると、右方向ヘシ

フトした場合には、符号を表す最上位ビットがシフトした所に入ってくるのです。もちろん、char型で宣言した数値0x80の最上位ビットは1ですから、(unsinged)宣言が必要だったというわけです。

このように、シフト演算子を使う場合は、使用する目的を十分に考慮すると同時に、変数を宣言する時には、符号の有無にも配慮しなければなりません。

物理的意味上し正は、 種類 伝数板の格をするすとかす ことばなります。

らも説明するまでもありませんが、メモリに格的する数値の初期値を

x80 (2進数では10000000となる)とし、X 座標を8で割った剰余で右方向

こシフトさせれば、正しいドット位置が設定できるというわけです。

なお、8で割った剩余は2進数の性質から、7との AND により求めること

ができます。ちょっとしたテクニックとして覚えておきましょう。

シフト演算子には、もうひとつ舞術的な意味があります。ご存じのよう

に数値の桁をシフトするということは、右方向にシフトさせればその数値 の基数で割ることを意味し、左方向にシフトすれば基数倍することになる

. *50

2進数の基数は2ですから、右方向に一回シフトさせれば2で割ることを窓

この性質を使って (X±8) を実現しているのが、LIST2-6.C

3)です。これは3回右方向にシフトするという意味ですから結果的に(X+

8) が求められることになるのです。なお、この海算は、単純に割り算する

よりもはるかに高速になります。

ところで、このシフト演算子を使う場合には、注意しなければならない 部分があります。まず、LIST2-6.Cで、初期値0x80を格納している変数 dot

set の宣言部分に着目してください。

(unsinged char)となっています。これは、この変数の数値の符号は考慮 とないことを言言しているのです。実は一符号を考慮すると、右方向へシ

■ 色の成分チェック

これでドットの位置は決定できますから、次に、色の指定について話しを進めることにします。LIST2-6.Cでは、指定した色がどのプレーンを使って表現されるのかという判断を、論理演算の&を利用して実現しています。では、このプロセスはどのように実行されているのかを考えてみましょう。

まず、色の種類は0~15でした。これらの数値は当然のことながら CPU は2進数として扱っています。これらを2進数で表現した数値を次に示しますから確認してください。

0		00000000	8		00001000
1	o to take o	00000001	9	/*******	00001001
2	nt xmax	00000010	10	/*	00001010
3		00000011	11	,	00001011
4	nt =2;	00000100	12	ner nebeli	00001100
5		00000101	13		00001101
6	witch (:	00000110	14		00001110
7	3-69-5	00000111	15	J(0)	00001111

CPU は、このように $0\sim15$ の色番号を2進数として認識するわけですが、ここでおもしろいのは、色を構成するプレーンの発色の有る無しと、色番号を表す数値のビットが1対1に対応しているということでしょう。すなわち、一番右側のビットが1であれば B プレーンが発色していることになり、右から2番目のビットが1であれば R プレーンが、同様に3番目は G プレーン、4番目は I プレーンが発色しているというわけです。

ですから変数 color に指定した色が、B プレーンを含むか否かのチェックは次の論理演算の結果をチェックすればよいことになります。

```
color \& 00000001 = 1 で B プレーンの発色有り color \& 00000001 = 0 で B プレーンの発色無し
```

同様に R, G, I プレーンは次のようになります。

color & 00000010 = 1 で R プレーンの発色有り

color & 00000010 = 0 で R プレーンの発色無し

color & 00000100 = 1 で G プレーンの発色有り

color & 00000100 = 0 でGプレーンの発色無し

| color & 00001000 = 1 で I プレーンの発色有り

color & 00001000 = 0 で I プレーンの発色無し

さて、ここでif文の動作を思い出してください。

条件式の評価は、(条件式 \neq 0) で真となり、(条件式=0) で偽となりますから、簡単に色の成分に従った処理ができるのです。すなわち、式の演算の結果が真(\neq 0) であれば対応するプレーンのビットをセットし、偽(=0) であれば対応するプレーンのビットをリセットするわけです。

5、一番石頭のピットが1であれば R ブレーンが明色していることになり、 5から2番目のビットが1であれば R ブレーンが、同様に3番目は G プレーン 4番目は I ブレーンが発色しているというわけです。

ですから変数 color に指定した色が、B ブレーンを含むか否かのチェッ

■ キーの入力を知るには

LIST2-6.C の関数は、任意の(X, Y)座標に指定した色でドット(点)を描き込む関数でした。そこで、この関数を利用して、キーの入力に従って次々と座標を変化させながらドットを打っていくことを実現させれば、どのような直線でも描けそうです。というわけで、キー入力に従って座標を更新する関数を定義してみましょう。

```
LIST2-7.C
extern int was to be the control of 
              y; \* x x x x + 0 除大量 0 聚基 x *\ /* y 座標格納用 */ (***** < *) **
xycset ( T S M O M O M M A M M - M * )
                                                                                                       /* x方向最小值 */
               int xmin,
                                                                                                                                     /* x方向最大值 */
               int xmax,
                                                                                                                                     /* y方向最小值 */
               int ymin,
              int ymax) /* y方向最大值 */
               switch(r2 = getch()) { /* キー情報に従って座標を更新する */ case '1':
   これで、キー情報に従って座標を更新する関数と、原:y+ 従ってドッ
                                             --x;
                                           break;
  ① 发出 case '2': 大主义立て 3 数型 mism 。 仓主 J 主意 4 3 计美心 1 8
                                           ++y;
                                           break:
                             case '3':
                                           ++v;
                                           ++x;
 A tobulo break; TTTTA A S 5 3 8 4 1
                             case '4':
                                             --x;
                                            break;
                             case '6':
                                            ++x:
```

```
break;
   case '7':
      --y;
     break:
   case '8':
   A(0) --v; 0 ) LH | | | |
      break;
   case '9':
   のような直線でも維持を密です。というをはこ。 明光:v=F 接近
     break:
   default: break:
if(y > ymax)
           v = vmax;
                     /* v座標の最大値のチェック */
if(y < ymin) y = ymin;
                     /* y座標の最小値のチェック */
if(x > xmax) x = xmax; /* x座標の最大値のチェック */
                     /* x座標の最小値のチェック */
if(x < xmin)
                     /* キー情報を関数の戻り値とする */
return(r2);
```

ここでは、TURBO C の関数 getch()でキーボードからの情報を得ています。そして、方向別の処理をする為に、テンキーの1, 2, 3, 4, 6, 7, 8, 9を対応させて、座標 (X, Y) の更新処理をしています。また、せっかく取得したキー情報は、関数の戻り値として返しています。

これで、キー情報に従って座標を更新する関数と、座標に従ってドットを表示する関数がそろいましたから、それぞれをリンクさせてコンパイルおよび実行といきましょう。main 関数とプロジェクトファイルは次のようになります。

```
LIST2-8.C
#include <graphics.h> /* 必要とされるヘッダファイルをincludeする */
extern void grphinit(void); /* 外部関数の宣言 */
extern void pointer_set(void);
extern void dot_plot(int);
extern int xycset(int, int, int, int);
```

```
void main (void)
                       /* main関数の定義 */
  int keydata = 0,
                       /* キー情報格納用 */
           = 639,
                       /* x座標の最大値 */
     xmax
   ymax
                       /* y座標の最大値 */
           = 399,
                       /* x座標の最小値 */
     xmin
     vmin = 0.
                       /* v座標の最小値 */
                       /* 描写する点の色の指定 */
  color
  grphinit();
                       /* グラフィックスシステムの初期化 */
                       /* ポインタの設定 */
  pointer set();
  while (keydata != 0x1b) {
                       /* [ESC] キーのチェック */
                       /* 点の位置およびキーデータ設定 */
   keydata = xycset(xmin, xmax, ymin, ymax);
     dot_plot(color);
                       /* 点の描写 */
 closegraph(); /* グラフィックスシステムの終了 */
```

LIST2-0 LIST2-1 LIST2-6 LIST2-7 LIST2-7

■ カーソルの表示

LIST2-8.PRJ を起動してみると、テンキーに従った直線は自由に引けるのですが、ドットを表示する現在位置が不明なのが非常に見づらいことが分かると思います。そこで次の課題として、現在のドット位置を示すカーソル表示へと話を進めることにしましょう。

まずカーソルの表示で考えなければならないのは、カーソルと、すでに表示されている絵との重ね合わせです。この時のカーソルはグラフィックスを描く為のカーソルですから、他の絵と同様にG.V.RAMへ表示しなければなりません。当然のことながら、カーソルを表示すればすでにG.V.RAM上に存在している絵は壊されてしまいます。カーソルを移動させると後には古いカーソルの残骸が残っているわけです。そこで、なんらかの工夫をして古いカーソルの部分を元の状態に復元することになります。

ここでは最も単純な背景との XOR (排他的論理和)をとる方法を試みてみます。次に XOR の真理値表を示しますから参照してください。

表2-4 XOR の真理値

X	Y	X (+) Y
0	0	0
0	1	1
1	0	1
1	1	0

(+): XOR

この真理値表をよくみるとXとYが異なれば演算結果が1となり、XとYが同じであれば、0となっていることがわかります。

今、カーソルのデータを X、背景の絵を Y とし、X と Y との XOR をほどこした演算結果を Z として考えてみましょう。

この場合、Z がカーソル表示によって破壊されたデータということになります。ここで問題となるのは、背景 Y を X との演算 XOR によって破壊したデータ Z をいかにして元の Y という背景のデータに復元するかとい

うことです。すなわち X と Z から Y が求まればよいわけですが、実は、これは XOR の性質から次のように簡単に求められるのです。

Y = X(+)Z

すなわち、カーソルのデータ X と背景 Y との X OR をほどこした部分 Z に、再び、カーソルのデータ X との X OR をほどこせば、元の背景 Y が復元されるというわけです。ですから、カーソルの表示、背景の復元、いずれの処理にしても単純に背景との X OR をとるプログラムのみで実現できるわけです。

```
LIST2-9.C
extern char far *c blue;
                           /* B面へのポインタの官言 */
extern char far *c green; /* G面へのポインタの宣言 */
extern char far *c_itsty; /* I面へのポインタの宣言 */
              /* カーソルのオン/オフの状態を示す */
int csr on = 0;
void cursorxor(
                           /* カーソルのメモリアドレス */
  int j,
                           /* カーソルのビットアドレス */
  int s,
   int c )
                           /* カーソルの色 */
  static unsigned int
                           /* カーソル形状データ */
  crsptn[] = {
     0x8000,
                           /* 10000000000000 */
      0xc000,
                           /* 11000000000000000
      0xc000,
                              11000000000000000
     0xe000,
                                           */
                              11100000000000000
      0xe000,
                           /* 11100000000000000
      0xf000,
                              11110000000000000
                           /*
     0xf000,
                           /* 11110000000000000
                                            */
                              11111000000000000
     0xf800,
                           /* 1111100000000000
                                            */ * * * * *
     0xfc00,
                              1111110000000000
      0xfc00,
                              1111110000000000
                                            */
      0xfe00,
                              1111111000000000
     Oxff00,
                                           */
                          /* 1111111100000000
      0x1800,
                             0001100000000000
```

```
0x1800, /* 000110000000000 */
               /* 00011000000000 */
     0x1800
  1:
  union {
                          /* カーソルデータのシフト用 */
    unsigned int crs;
     unsigned char crs2[2];
  ) pt; 5 3 1 4 4 4 0 X ( - 1
                          /* 使用変数の宣言 */
 csr_on ^= 1; /* カーソルの表示状態を示す */
  for (i = 0; i < 16; ++i, j+=79) {
     pt.crs = crsptn[i] >> s;
 if(c & 1) *(c_blue + j) ^= pt.crs2[1];
     if(c & 2) *(c red + i) ^= pt.crs2[1];
if(c & 4) *(c green + j) ^= pt.crs2[1];
if(c & 8) *(c itsty + j) ^= pt.crs2[1];
     ++j;音音のもくトルの小面目
     if(c & 1) *(c blue + j) ^= pt.crs2[0];
     if(c & 2) *(c red + j) ^= pt.crs2[0];
     if(c & 4) *(c green + j) ^= pt.crs2[0];
  if(c & 8) *(c itsty + j) ^= pt.crs2[0];
```

カーソル形状データとしては、ドットの状態を2進数に換算して、さらに、 それを16進数に変換したものを直接データとして持たせています。原始的 なデザイン方法ですがカーソル程度であればこれでも十分といえるでしょ う。参考までに2進数による表記をコメントとして載せてあります。

また、TURBO C では2進数が扱えない為に16進数としましたが、問題となるのは2進数のデータと16進数のデータとの対応でしょう。これには専用の電卓(2進演算のできるもの)などがあると便利ですが、もし無い場合には4ビットずつ区切ることによって表1-1から以外と簡単に16進数へと変換できます。

カーソル形状データはメモリの記憶単位の二倍の int 型とし、シフト演算によってドット単位の移動を実現しています。この時、配列の定義として unsigned 宣言が必要となることに注意してください。

ところで、int型の数値は下位バイト、上位バイトの順でメモリへ格納されるのですが、この場合、メモリの若い方が画面左側に位置していることから処理としては工夫が必要になってきます。すなわち、このままメモリに書き込んだ場合には、上位バイトと下位バイトの左右の位置関係が逆転することになるわけです。

そこで、一旦シフトしたカーソルデータを共用体として定義した変数を 介してバイト単位の数値に分割してから上位バイト、下位バイトという順 番で格納しています。以上のことを踏まえて、これらを組み合わせたプロ ジェクト LIST2-10を実行してみましょう。

```
tx & t 3 May e stal LIST2-10.C ) (sysbyex) dodler
                         /* 必要とされるヘッダファイルをincludeする */
#include <graphics.h>
#include <bios98.h>
#define ON 1:
#define OFF 0:
extern void grphinit (void); ____ (1 + glmp x rolop) 11
extern void pointer set (void);
extern void dot plot(int);
extern int xycset(int, int, int, int);
extern void cursorxor(int, int, int);
extern int
                       /* x座標格納用 */
   x.
                       /* v座標格納用 */
   У,
   memory adr,
                        /* メモリアドレス格納用 */
                        /* ビットアドレス格納用 */
   bit adr,
                       /* カーソルの現在のon/off状態を示すサイン */
   csr_on,
                        /* カーソル表示の管理用サイン(1でon, 0でoff) */
   sign;
void main()
                           /* キー情報格納用 */
  KEY INFO kinf;
                           /* キーデータ保存用 */
   int kevdata = 0,
       xmax
            = 639,
                           /* x座標最大值 */
                           /* y座標最大值 */
            = 399,
       ymax
       xmin
              = 0,
                           /* x座標最小值 */
       vmin
              = 0,
                           /* v座標最小值 */
```

```
sfsin = 0, /* シフトサイン用 */
 cmax = 15, /* カラー番号の最大値 */
          = 0,
                    /* カラー番号の最小値 */
    cmin
color = 1: /* 現在選択されているドットカラー */
  grphinit(); /* グラフィックスシステムの初期化 */
                    /* 各プレーンへのポインタの設定 */
  pointer_set();
  cursorxor(memory_adr, bit_adr, color); /* カーソルの表示 */
 while(keydata != 0x1b) {
 keydata = xycset(xmin, xmax, ymin, ymax); /* キー情報を取得 */
    kinf.cmmd = 2; /* bios98keyへのコマンドの設定 */
    sfsin = bios98key(&kinf); /* シフトキーの情報を取り込む */
    if(csr_on == 1) { /* カーソルサインがonならばカーソルの消去 */
    cursorxor (memory adr, bit adr, color);
    switch(keydata) { /* キー情報に従って処理をする */
 A Tabuta case 'x':
    case 'X':
    if(color < cmax - 1) ++color;
     break; weap + 11 - pt.cma2101; ... 11. NO onlineby
    case 'z': seety a 13 cm pt.ore2[0]:
       case 'Z':
          if(color > cmin + 1) --color; (ov) standary blow missing
       break:
      defaelt : break;
if(keydata!='') { /* [space]キーが押されていればカーソルを消去 */
       if(sfsin & 1) sign = 0;
       else
                  sign = 1;
dot plot(color);
       cursorxor(memory_adr, bit_adr, color);
                  /* グラフィックスシステムの終了とする */
  closegraph();
```

```
LIST2-10.PRJ
```

```
LIST2-0
LIST2-1
LIST2-6
LIST2-7
LIST2-9
LIST2-10
```

どうですか、カーソルが表示されたでけでもグレードがグッと上がったのではないでしょうか。実際にプロジェクト LIST2-10.PRJ を走らせて、いろいろな直線を描いてみてください。また、X, Z キーで色の変化を、スペースキーでカーソルの ON/OFF を管理するようにしてありますから、これらも同時に試してください。LIST2-10.C まで進んでくると、G.V. RAM の構成や仕組みがかなりハッキリとしてきたのではないでしょうか?

たまを様式制しなある機理を実信をている時になり外対を主動権出社だ こか、どこかから通信を受けたとか、割り込みは様々な要因で発生します。 では、はこれもの要因を確立しているがのの「電子」をサーチし、要因 例の処理へと制御を移び助きする。そして、側り込み処理が終了すれば、ま で元の処理へと制御を移む脚をを設置していますます。 これらの処理は、高速に行われるために人間にします。 これらの処理は、高速に行われるために人間にします。 これらの処理は、高速に行われるために人間にします。 これらの処理は、高速に行われるために人間にします。 これらの処理は、高速に行われるために人間にします。 これらの処理は、高速に行われるために人間にします。 これらの処理は、まるに、割り込みを高調させない所に高差があるのです。

なを使う場合には各々のドライバのマニュアルを参照してください。 とでは マウスからの信頼域割り込みとして処理 選択をすがきつウスドライバを 使っている限り、特に割り込みに関する知識は要求されません。また、マスドライバを使うといっても、直接使うわけではなく、TURBOCに行 薬の関数を介し、間接的に使うことになります。では、次のリストを打ち 込んでください。

LIST2-11.C

図2-6 割り込み処理の実行過程

onern char far *c_blue; /* char超過国へのポインタの宣誓 * atexn char far *c_red; /* char覆を図へのポインタの宣言 *

where ober far to itsty: /* char器IBへのポインチの質質

■ 割り込み処理の実際

次のチャレンジとしては、ゲームを作る上で欠かせない割り込み処理の 方へと話しを進めていきましょう。

割り込みと聞くと、あまり良いイメージがありませんが、コンピュータの世界では、欠かせないものなのです。

たとえば、CPUがある処理を実行している時にストップキーが押されたとか、どこかから通信を受けたとか、割り込みは様々な要因で発生します。 CPUはこれらの要因に従って、割り込みベクタテーブルをサーチし、要因別の処理へと制御を移すのです。そして、割り込み処理が終了すれば、また元の処理へと制御を移し粛々と処理を続行していきます。

これらの処理は、高速に行われるために人間にとっては割り込まれたというよりも、あたかも並列に処理が進行していくように思えます。割り込みとは、まさに、割り込みを意識させない所に意義があるのです。

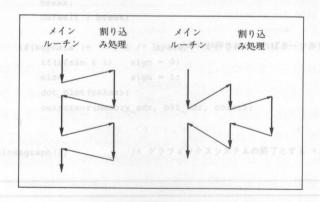


図2-6 割り込み処理の実行過程

■ マウスを組み込めば、お絵書きツールのでき上り

では、割り込み処理の入門として、ここではマウスを取り上げてみましょう。マウスの割り込み処理を利用することによって、プロジェクトLIST2-10.PRJをお絵書きツールへと変身させようというわけです。

まず、マウスを使う為には、MS-DOSのデバイスドライバにマウスドラバを組み込まなければなりません。手順としては次のようになります。

手順1	CONFIG. SYSへMOUSE. SYSを登録		DER INFO
手順2	MS-DOSシステムを再起動	k = 8 /	0382_860
手順3	マウスシステムの初期化	(i) Jol	id madots
手順4	マウス環境の設定		

(水) 簡單

単語体定義 */ sold medatalon

> int status int lft_bu int rgt bo

図2-7 マウスドライバの組み込み手順

本書では、NEC版マウスドライバに対応させていますから、他のドライバを使う場合には各々のドライバのマニュアルを参照してください。

マウスからの信号は割り込みとして処理されますが、マウスドライバを使っている限り、特に割り込みに関する知識は要求されません。また、マウスドライバを使うといっても、直接使うわけではなく、TURBOCに付属の関数を介して間接的に使うことになります。では、次のリストを打ち込んでください。

```
#include <bios98.h> /* 必要とされるヘッダファイルをincludeする */
#include <dos.h>

extern char far *c_blue; /* char型B面へのポインタの宣言 */
extern char far *c_red; /* char型R面へのポインタの宣言 */
extern char far *c_green; /* char型G面へのポインタの宣言 */
extern char far *c_itsty; /* char型I面へのポインタの宣言 */
```

```
extern void dot_plot(int);
                            /* ドット描写用外部関数の宣言 */
extern void cursorxor(int, int, int); /* カーソル表示用外部関数の宣言 */
extern int
                            /* 外部変数の宣言 */
                            /* x座標格納用 */
                            /* y座標格納用 */
                  /* 指定色格納用 */
   color.
                            /* カーソルの表示または消去を示す */
   bit adr,
                            /* ビットアドレス */
  csr on, /* カーソルの表示状態を示す */
                    /* メモリアドレス */
   memory_adr;
MOUSE INIT msint; /* マウス初期化用構造体定義 */
union REGS reg;
                            /* レジスタ用構造体定義 */
MOUSE INFO msinf:
                            /* マウス情報格納用構造体定義 */
void msdotplot(int, int, int, unsigned, unsigned);
void mouseinit (
  unsigned int mxmin.
                            /* マウスの可動範囲x方向の最小値 */
   unsigned int mxmax,
                            /* マウスの可動範囲x方向の最大値 */
 unsigned int mymin,
                            /* マウスの可動範囲v方向の最小値 */
  unsigned int mymax )
                            /* マウスの可動範囲y方向の最大値 */
  msint.cmmd = 0; /* コマンドを初期化とする */
  bios98mouse init(&msint);
                            /* マウス環境の初期化 */
  msint.cmmd = 12;
                            /* コマンドを関数の登録とする */
  msint.call cond = 0x1f; /* msdotplotを呼び出す条件 */
                         /* 関数名msdotplotを登録 */
  msint.mouse fun = msdotplot;
  bios98mouse init(&msint);
                        /* msdotplotをユーザー定義関数とする */
  msint.cmmd = 16;
                 /* コマンドをx方向可動範囲の設定とする */
  msint.min h pos = mxmin; /* x方向の可動範囲の最小値を指定する */
  msint.max h pos = mxmax;
                       /* x方向の可動範囲の最大値を指定する */
  bios98mouse init(&msint);
                        /* 可動範囲の設定 */
  msinf.cmmd = 4;
                     /* コマンドをマウスカーソル位置の設定とする */
  msinf.abs h crs = mxmin;
                        /* マウスカーソルのx方向初期値を設定 */
                     /* マウスカーソルのy方向初期値を設定 */
  msinf.abs_v_crs = mymin;
  bios98mouse(&msinf); /* マウスカーソルの初期位置の設定 */
  x = mxmin;
                        /* ドット描写用x座標の初期化 */
  y = mymin;
                       /* ドット描写用v座標の初期化 */
  bit_adr = mxmin & 7;
                       /* ビットアドレス初期化 */
  memory adr = (mxmin >> 3) + mymin * 80; /* メモリアドレス初期化 */
```

```
void mousexyset(int x0, int y0) /* マウスカーソルの位置を設定する */
                       /* コマンドをマウスカーソル位置の設定とする */
  msinf.cmmd = 4;
msinf.abs_h crs = x0; /* x方向位置をx0とする */
                      /* y方向位置をy0とする */
  msinf.abs v crs = y0;
                       /* マウスカーソル位置を座標(x0, y0)とする */
  bios98mouse(&msinf);
int msonoff = 1; /* マウスのオン/オフ管理用 */
                       /* マウスシステムの終了とする */
void mouseterm (void)
NO.4…Y 座標のとりうる優大値 setlent
                       /* コマンドを初期化とする */
  msint.cmmd = 0;
  bios98mouse_init(&msint); /* マウスシステムの初期化 */
                       /* マウス用ユーザー定義関数 */
void msdotplot (
                       /* マウスの状態 */
   int status,
                       /* マウス左ボタンの状態 */
   int lft buton,
                     /* マウス右ボタンの状態 */
   int rgt buton,
                      /* マウス用カーソルの水平軸座標 */
   unsigned abs h crs,
                      /* マウス用カーソルの垂直軸座標 */
   unsigned abs v crs )
if(msonoff == 1) { /* msonoff=1ならカーソル表示し座標を更新 */
      if(csr_on == 1) { /* カーソルが表示状態であればカーソルを消去 */
         cursorxor (memory adr, bit_adr, color);
     在沙水学等级押告相飞
                       /* x軸カーソル位置を新たにx座標とする */
      x = abs_h_crs;
                    /* v軸カーソル位置を新たにy座標とする */
      y = abs v crs;
      if(rgt_buton == -1) { /* 右ボタンがオンであればカラーを変更する */
         if (status & 8)
        ++color;
        if(color == 16) color = 0;
      if(lft_buton == -1) sign = 1; /* 左ボタンがonなら点の描写とする */
      else
                     sign = 0;
      dot_plot(color); /* signに従って点を描写する */
      cursorxor(memory_adr, bit_adr, color); /* カーソルを表示する */
   "はなく、マウス割り込みによって起動される関数として登録しまけ
   else if(csr_on == 1) ( /* カーソルが表示されていればカーソルの消去 */
      cursorxor (memory adr, bit adr, color);
   うだしてありますので、マグヴァキリ
```

LIST2-11.C ではマウスに関する基本的な関数を用意しました。各関数の機能は次のようになっています。

mouseinit … 【機能】 マウスシステムの初期化及び環境の設定

【引数】 NO.1…X 座標のとりうる最小値

NO.2…X座標のとりうる最大値

NO.3…Y座標のとりうる最小値

NO.4…Y座標のとりうる最大値

mousexyset …【機能】 マウスカーソルの表示位置の設定

【引数】 NO.1…カーソルの X 座標

NO.2…カーソルのY座標

mouseterm …【機能】 マウスシステムの終了

msdotplot … 【機能】 マウスの割り込みによって起動される関数

で、カーソルの表示と、右のボタンが押されていれば、カーソル位置にドットをプロットする。また、左のボタンが押されていれば、

カラーを変化させる。

【引数】 NO.1…割り込みの要因

NO.2 左ボタンの状態 (0:解放, -1:押下)

NO.3 右ボタンの状態 (0:解放, -1:押下)

NO.4···X 座標

NO.5…Y座標

ここで中心となる関数は msdotplot()ですが、これは、直接コールするのではなく、マウス割り込みによって起動される関数として登録します。すなわち、この関数が割り込み処理というわけです。また、カーソルはこの関数 msdotplot()で表示するようにしてありますので、マウスドライバ独自のカーソルは OFF としてあります。カーソルの表示及びドット表示の

プロセスは今までのものと変わりがありません。

さて、プロジェクト LIST2-10に LIST2-11を関連させたものが、プロジェクト LIST2-12.PRJ です。

```
LIST2-12.C
#include <graphics.h> /* 必要とされるヘッダファイルをincludeする */
#include <dos.h>
#include <bios98.h>
#define ON 1
extern void grphinit(void); /* 外部関数の宣言 */
extern void pointer_set(void);
extern int xycset(int, int, int, int);
extern void cursorxor(int, int, int);
unsigned int, 44 #4 Testiliple #\ (10000) tolg tob
unsigned int, (rolos abs sid abs viomem) ioxiosius
  unsigned int,
  unsigned int
);
extern void mouseterm(void);
extern void mousexyset(int, int);
               /* 外部変数の宣言 */
extern int
               /* x座標格納用 */
  x,
               /* y座標格納用 */
  y,
              /* メモリアドレス */
  memory_adr,
               /* ビットアドレス */
  bit adr,
               /* カーソルの表示状態を示す */
  csr on,
               /* カーソル表示の管理用サイン(1でon, 0でoff) */
  sign;
int color = 4:
void main()
                   /* キー情報格納用構造体定義 */
  KEY INFO kinf;
  int sfsin, /* シフトキー情報 */
   keydata = 0, /* キーデータ管理 */
                   /* x座標最大值 */
     xmax = 639,
```

```
/* v座標最大值 */
vmax = 399
                     /* x座標最大値 */
    xmin = 0,
    ymin = 0;
                     /* v座標最小值 */
                     /* グラフィックスシステムの初期化 */
 grphinit():
                    /* 各プレーンへのポインタの設定 */
 pointer set();
 mouseinit(xmin, xmax, ymin, ymax);
                            /* マウスシステムの初期化 */
 cursorxor(memory adr, bit adr, color); /* カーソルの表示 */
                             /* 機能コードセット */
 kinf.cmmd = 2:
 while(keydata != 0x1b) {
                             /* [ESC] キーチェック */
    keydata = xycset(xmin, xmax, ymin, ymax); /* キー情報セット */
    sfsin = bios98key(&kinf); /* シフトキー情報の取り込み */
                    /* 割り込み禁止 */
    disable():
    if(csr on == ON) {
                    /* カーソルサインがonならばカーソル消去 */
 cursorxor(memory adr, bit adr, color); /* カーソルの消去 */
                    /* マウスカーソル位置の指定 */
   mousexyset(x, y);
    if (kevdata!= '') { /* [space] キーが押されているかのチェック */
       sign ^= 1; /* シフトキー情報に従ってドットのon/offを決定 */
      dot_plot(color); /* signに従ってドットをプロットする */
      cursorxor(memory adr, bit adr, color); /* カーソルの表示 */
 enable();
                  /* 割り込みの許可 */
                    /* マウスシステムの終了 */
 mouseterm();
                  /* グラフィックスシステムの終了とする */
 closegraph();
```

LIST2-12.PRJ

```
LIST2-0
LIST2-1
LIST2-6
LIST2-7
LIST2-9
LIST2-11
LIST2-12
```

ここで注意するのは、テンキーによって座標が変更される場合です。というのも、マウス割り込みによって起動される関数 msdotplot()でも座標を更新しているために、関数どうしが競合してしまうという問題が生じる

からです。

この競合という問題を避けるために、テンキーによる一連のプロセスに入る前で割り込みを禁止(disable)し、処理の終了後、割り込みを許可(enable)しています。すなわち G.V.RAM に対する処理の一本化を図るわけです。このへんの所が割り込み処理の面倒な所とも言えるでしょう。

実際に実行してみると、プロジェクト LIST2-12.PRJ は、お絵書きツールの基本形になっていることがわかると思います。すなわち、これに色々な飾りを付けると、市販のツールとなるわけです。興味のある方は、ぜひチャレンジしてみてください。

これで第2章は幕となります。次の第3章では、第2章の応用として、TV ゲームには必須のツールである。パターンエディタというプログラム作り にチャレンジすることにしましょう。

```
vmax = 399.
关系前专翻像最级必禁患 (disable)、U、処理の終了後、調力込命を許可
        She G. V. RAM EXIT & CHILD
わけです。」に初かぬんの所が割り込みが、悪の飛倒治所と幸福するをではまう。
     るだ、ナロジェクト LIST2 12. PRI は、朝候
いの基本形態なうでいる。ことがあかる。と思います。これなり物であれた色々
    it (car_on == ON) ( パーカーソルさんを含むなり様でする
                   adr. bit adr. opior);
こんで第2章は幕となります。次の第3章では、第2章の応用として、TV
ムには必須の美口気がである。スペラニンエディのとはキャヤログラム作り
     if (keydsta != ' ') { - /* lapace | キーが押されているかのチェック *)
        aign = stain & 1: /* population and to a stain a grant
        sign ** 1: /* シフトキー情報に従ってギットのon/oをfを決定 */
        dot plot(color); /* signに従ってドットをプロットする */
        cursorsor(memory adr, bit adr, color); /* カーソルの表示 */
     anable();
                    /※ 願り込みの許可 */ ...
                    /* マウスシステム的終了 *ナ
   mouseterm();
                    フェ グラフィックスシステムの終了とする モルー
   closegraph();
```

LIST2-12.PRJ

LIST2-0 LIST2-1 LIST2-6 LIST2-7 LIST2-9 LIST2-11

LIST2-12

ここで注意するのは、テンキーによって座標が変更される場合です。と いうのも、マウス割り込みによって起動される関数 rasdotplot () でも座標 を要新しているために、関数とうしが総合してしまうという問題が生した

第3章 パターンエディタに挑戦

本章で取り上げるパターンエディタは、TV ゲームにはなくてはならない必須のツールと言えるものです。しかも、パターンエディタにはTV ゲームで使われる基本的なテクニックが山ほどありますから、ウォーミングアップのつもりで取り組んでください。

もちろん、本書で作りあげるゲーム用のキャラクタをデザインする為にも使いますし、その他にも色々と利用価値のあるツールとなります。では、汎用 BOX ルーチンへと話を進めることにしましょう。

DISTS-0.0

は513-0.しのプログラムの挑手は、ドットを入力をは最高を発展して前としています。ですから、2章で並

atem obsr fer for the state of the shade of the state o

マロス Constitution (人は、グラフィンの主義をしている。 Characon Constitution (人は、グラフィンのまつ Constitution (人は、グラフィンのよう Constitution (人は、からないのよう Constitution (人は、グラフィンのよう Constitution (人は、からないのよう Constitution (人は、からないのよう Constitution (人は、からないのよう C

vramdoteet(
int memory_adr, /* 表示アドレス */ でランドンドル

の関数の自由度は立て2寸音でします。また、G.V.RAMedia

の変数 dot 現まです色を管理している変数 colorに by applifit ashift tanit planed,

っている所にも連盟して内できた。では、LIST3-0、Cを連絡を対策し

int dshift) /* 表示デーサジフト用 */

cher dot rest /* 7-29ty - ff *

in i ani

■ BOX ルーチンの作成

点の集まりは直線になり、直線の集まりは面となるわけですが、TV ゲームに出てくるキャラクタは基本的に BOX 型をしています。複雑に見えるパターンであっても、BOX 状に並んでいるデータをくり抜くことによって表現しているのです。しかも、X 方向はしばしばメモリの記憶単位である8ドットを基本構成とし、Y 方向も、X 方向に合わせて8ドット毎にデザインすることが多いようです。すなわち、縦横8ドットが、パターンをデザインする時の基本単位となるわけです。

ここでは、任意のG.V.RAM アドレスに、任意の色、任意の大きさで、BOX 状に描写する関数を作ってみましょう。では、LIST3-0.C です。

```
LIST3-0.C
#include <stdlib.h>
extern char far *c blue;
                                /* char型B面へのポインタの宣言 */
                                /* char型R面へのポインタの宣言 */
extern char far *c red;
                                /* char型G面へのポインタの宣言 */
extern char far *c green;
extern char far *c itsty;
                                /* char型I面へのポインタの宣言 */
void vramdotset (
   int memory adr,
                                /* 表示アドレス */
   int color,
                                /* 表示カラー */
                                /* 表示データ */
   char dot_set,
                                /* Bプレーンチェック用 */
   int planeb,
                                /* Rプレーンチェック用 */
   int planer,
                                /* Gプレーンチェック用 */
   int planeg,
                                /* エプレーンチェック用 */
   int planei,
                                /* x方向ループ回数 */
   int loopx,
                                /* v方向ループ回数 */
   int loopy,
                                /* 表示カラーシフト用 */
   int shift,
                                /* 表示データシフト用 */
   int dshift )
                                /* データリセット用 */
   char dot res;
   int i, j;
```

```
for(i = 0; i < loopy; ++i) { /* Y方向のループ */
   dot res = ~dot set; /* リセット用データを求める */
   for(j = 0; j < loopx; ++j) { /* x方向のループ */
    if(color&planeb) *(c blue + memory adr + j) |= dot set;
       else *(c_blue + memory_adr + j) &= dot_res;
       if(color&planer) *(c red
                              + memory adr + j) |= dot set;
                      *(c red + memory adr + j) &= dot res;
       if(color&planeg) *(c green + memory adr + j) |= dot set;
                      *(c green + memory adr + j) &= dot res;
       if(color&planei) *(c itsty + memory adr + j) |= dot set;
       else
                      *(c itsty + memory adr + j) &= dot res;
                            /* 次ラインの表示アドレスを求める */
   memory adr += 80;
   dot_set = rotl(dot_set, dshift); /* 表示データをシフトする */
   color <<= shift;
                            /* 構成色のチェック用データを求める */
```

LIST3-0.Cのプログラムの流れは、ドットをX方向に展開して線を、さ らに、その線を Y 方向に展開して面としています。ですから、2章で組んだ ドット描写用の関数を、少し改造するだけで完成です。

特にこの関数で工夫した所といえば、グラフィックスの色を構成してい るプレーンを判断するためのデータ (planeb~i) を、引数として可変にし たということです。

このように、プレーンを選別するためのデータが可変になるだけでも、 この関数の自由度はかなりアップします。 また、 G. V. RAM へ書き込むた めの変数 dot set と、色を管理している変数 color にも、dshift、shift と いう引数を作用させてシフトすることが可能(=0ではシフトされない)と なっている所にも注意してください。では、LIST3-0.Cを実際に利用して、 画面左上に赤い色で四角形を描いてみましょう。

```
list2-0
list2-1
list3-0
list3-1
```

さて、パターンエディタではドット単位で編集するわけですから、ドットの状態を的確に把握しなければなりません。しかし、ドットが細かいために、画面から直接ドットの状態を把握することはほとんど不可能です。そこで、このドットの状態を1ドット毎に拡大表示したいという欲求が生じるわけです。

ここで、先ほどの LIST3-0.C でグラフィックスの色を構成するプレーンを判断する為のデータ (planeb~i) を、なぜ、引数としなければならないのかという理由が分かったのではないかと思いますが、次の関数では、

G.V.RAM上のデータを読み込んで、色を構成するプレーンを判断する為の情報としています。これをビット単位でチェックすることによって、簡単にG.V.RAM上に展開したグラフィックスを拡大表示できるわけです。では、このあたりの所に注意して次のLIST3-2.Cへと進んでください。

```
LIST3-2 C
#include <graphics.h>
#include <comio.h>
extern void vramdotset (int, int, char, int, int, int,
                  int, int, int, int, int);
extern char far *c blue;
extern char far *c red;
extern char far *c green;
extern char far *c itsty;
extern int xmax; /* 編集エリアのX方向最大値 */
extern int ymax; /* 編集エリアのY方向最大値 */
int wwindow = 1, /* 編集エリアのG.V.RAMアドレス */
kwindow = 1 + 40 * 80 * 2 + 80 * 48;
                   /* 拡大表示エリアのG.V.RAMアドレス */
/* 編集エリアを拡大表示する */
void zoombitdt (void)
int bitb, bitr, bitg, bitl, bitt, i, j, k, m, n, o, p;
   char bdtb, bdtr, bdtg, bdtl;
   k = kwindow + 80; (# 9 (be > 8) exis xod sugnlaW") linking
   m = wwindow;
   p = xmax / 8; ndow, 2, 0xff, 1, 8 / (8 \ (1 + xamx)) - xamx
   for(o = 0; o < ymax; ++o) {
   for(n = 0; n < p; ++n) {
   getch ( bitt = 0x80; "myx**** rorre jugal *****av") liging
   bitb = *(c blue + m);
         bitr = *(c red + m);
         bitg = *(c green + m);
         bitl = *(c itsty + m);
```

```
0 (1817 ++m: 4 ) 0 (3)
  for(j = 0, i = 0; j < 8; ++j) {
            if(i & 1) vramdotset(k++, bitt, 0x0e, bitb, bitr,
  bitg, bitl, 1, 3, 0, 0);
            else vramdotset( k , bitt, 0xe0, bitb, bitr,
                              bitg, bitl, 1, 3, 0, 0);
bitt >>= 1;
#include. <bips98,h>
      m += 80 - p;
      k += 80 * 4 - xmax / 2;
extern vold vreyant acht trai Wancheni What Whet Miner blov nreme
/* 拡大エリアの枠の描写 */
void dispwaku (void)
   int x0, y0;
  y0 = 16 * 8 - 2:
  line(x0, y0, xmax * 4 + 9, y0);
  line(xmax * 4 + 9, y0, xmax * 4 + 9, y0 + ymax * 4 + 3);
   line (xmax * 4 + 9, y0 + ymax * 4 + 3, x0, y0 + ymax * 4 + 3);
  line(x0, v0 + vmax * 4 + 3, x0, v0);
) list2-0
/* 編集範囲の設定 */
void initset (void)
   while(1) {
      clrscr(); 749 THE VARIOUS CREATED HOTOLOGIC
     printf("Yninput box size (8 < 64) ? "); 108 + wobalws or x
      scanf("%d", &xmax);
    xmax = ((xmax + 7) / 8) * 8;
      if(xmax <= 64 && xmax >= 8) break;
printf("\n**** input error ****\n");
       先ほどの LIST3-0.C でダネマイ Bag Aの色変構成するアレ
   vmax = xmax;
1を判断する為のデータ (planeb~i) 係 型理、引動Pせなければなら
```

この LIST3-2.C では、拡大表示する関数と、編集の範囲を8~64ドットで設定する関数、そして、拡大エリアを見やすいように線で囲む関数を用意しました。

さっそく、これらの関数を組み合わせたプログラムを起動してみましょ う。メイン関数とプロジェクトファイルは次のようになります。

```
LIST3-3.C
  を的確に把握することが可能となったわけです。
拡大表示のテスト
*ケは煮こちないものとなってしまうでしょ
#include <graphics.h>
#include <dos.h>
#include <bios98.h>
extern void grphinit (void);
extern void pointer set (void);
extern void vramdotset (int, int, char, int, int, int,
               int, int, int, int, int );
extern void zoombitdt(void);
extern void dispwaku (void);
extern void initset (void);
extern int wwindow, kwindow;
int xmax, ymax;
void main()
  grphinit();
  pointer_set();
  initset();
  vramdotset(wwindow, 2, 0xff, 1, 2, 4, 8, xmax / 8, ymax, 0, 0);
 dispwaku();
  zoombitdt();
          Faに書き込まれている割り込みペクタを参照し、所定
  getch();
closegraph();
```

LIST3-3.PRJ

list2-0

list2-1

list3-0

list3-2

いかがでしたか、少しはパターンエディタらしくなりました。これで編集したドットの状態を的確に把握することが可能となったわけです。

<d animaras abataa)

finglude <dos.h>

<0.868010> epulonia

estern void grphinit (void);

extern void vramdotset(int.

int, int

(blow) Distances blow master

mx * 4 + 9, y0 + ymax * 4 + 3, xb(blow), textint blow gratica

extern int wwindow, kwindow;

The support of all the second of the second

round frames here

Unlaw blow

grphinic();

pointer_set()

initaet();

dispwaku();

dispwaku();

() dosep

closegraph();

CONTRACTOR AND ADDRESS.

割り込み処理の実際(点滅カーソルの表示)

さて、カーソル表示は前にもやりましたが、パターンエディタ用のカー ソルとしては、もっとカーソルらしく等間隔に点滅するカーソルを作って みたいものです。一口に等間隔といっても、処理としては以外と複雑になっ てしまいます。たとえば、プログラムで適当なウェイトをかけて、表示、 点滅を繰り返す方法等が考えられますが、これでは、プログラムの動きと してはぎこちないものとなってしまうでしょう。

そこで、等間隔でタイミングをとる割り込みを利用することになります。 まず、プログラミングを行う前に、割り込みにはどの様なものがあるかを 次の表で参照してみてください。

表3-1 PC-9801シリーズ割り込みベクタ

0	除算エラー
1	シングルステップ割り込み
2	NMI メモリパリティエラー
3	ブレークポイント割り込み
4	オーバーフロー割り込み
5	COPY キー割り込み
6	STOP キー割り込み
7	インターバルタイマ割り込み
8	タイマ割り込み
9	キーボード割り込み
10	V-SYNC 割り込み
k.ee	A&Ofol) () / / / / / / / / / / / / / / / / / /

以上ですが、CPU は、これら割り込みの要因に従って、メモリの0000: 0000H~0000:03FFHに書き込まれている割り込みベクタを参照し、所定の

ですから、これらの割り込みのうち等間隔に発生する割り込みを利用す ることになります。ここでは10番目の V-SYNC 割り込みを利用して等間 隔に点滅するカーソルを作ってみます。V-SYNC割り込みとは、1/60秒毎の CRT の垂直帰線区間にかかる割り込みのことです。

ところで、マウスの割り込み処理の時には、TURBOCの関数によって 自動的に煩わしい一連の処理を済ませてしまいましたから、とても楽でし たが、実際には次のような手順となります。

手順1 …interrupt 修飾子を用いて割り込み用関数を定義する

手順2 …対応する割り込みテーブルの元々のベクタと割り込みマスク レジスタの状態を保存する

手順3 …対応する割り込みテーブルへ、目的の関数へのベクタを登録 する

手順4 …割り込みマスクレジスタの割り込みに対応するビットをクリアして割り込みを有効にする

手順5 …処理が終了する時には、割り込みベクタテーブルと割り込み マスクレジスタを元の状態に戻す

では、これらの手順に従って割り込み関数を定義してみましょう。

Eとになります。ここでは10番目のV-SYNC割り込みを利用用(事等)

```
kaku adr = 0,
   color sad = 2, is int to int k) t(S) discount w incomes
   dot adr = 0,
   half = 0,
   crson = 0 = 1,  k) ++1, i+=60) (
   crssign = 0,
   crsinf = 1, (1) % j
   space;
unsigned char dot set = 0x80;
void dotcrs (void);
int counter = 0,
   counter2 = 0;
/* 点滅カーソル割り込み関数の定義 */
void interrupt idotcsr(void)
   char i;
   ++counter2:
   if(crsinf == 1) {
      if(++counter == 30) counter = 0;
      if (counter > 15) crson = 0; ( == aplease as 0 == acero) 11
      else de crson = 1;
    dotcrs(); ) (0 -- npissio 22 1 -- nosio) il esle
   else crson = 0; (E << xe) + 08 * ve + wobniww = abs job
   i = 0x20; 10 * 08 * 00 + (1 << mo) + wobniws = mbs mass
   outportb(0, i);
   outportb(0x64, i); see job abs job alss abs wish) resums
char keeport;
void interrupt (*keepvector)();
void interrupt idotcsr(void); abs dob dal as dat a dal) rosano blov
/* 点滅カーソルの初期設定 */
void initcsr(void)
   dot_set = 0x80 >> (ex & 7); (8 .00x0 .x) sbessiviox esis
   dot adr = wwindow + ey * 80 + (ex >> 3);
   kaku adr = kwindow + (ex \gg 1) + ey * 80 * 4;
```

```
half = ex & 1;
  keeport = inportb(2);
  keepvector = getvect(10);
  setvect(10, idotcsr);
  disable();
  outportb(2, keeport & 0xfb);
  enable();
  outportb(0x64, 1);
/* 点滅カーソルの終了 */
void termcsr(void)
 setvect(10, keepvector);
  outportb(2, keeport);
  enable();
/* カーソルの描写及び消去 */
void dotcrs (void)
  if(crson == 0 && crssign == 1) cursor( kaku_adr, half,
    dot_adr, dot_set );
  else if (crson == 1 && crssign == 0) {
    dot set = 0x80 >> (ex&7);
    dot_adr = wwindow + ey * 80 + (ex >> 3);
   kaku_adr = kwindow + (ex >> 1) + ey * 80 * 4;
    half = ex & 1:
   extern int kwindow;
wtern char far *c blue;
/* カーソルの描写 */
void cursor(int k, int s, int dot adr, int dot set) addressed blow
 crssign '= 1;
  if (s == 0) xorvramdt (k, 0xe0, 5);
  else xorvramdt(k, 0x0e, 5); ( 3 xs) << 08x0 = 388 30b
```

```
/* G.V.RAMへ指定データを書き込む */
void xorvramdt (int i, int j, int k)
(stern woid initestive(d)
 int 1;
  for(1 = 0; 1 < k; ++1, i+=80) { (reotobi) , i > 0.1-8TSLI
     *(c blue + i) ^= j;
*(c green + i) ^= j;
 *(c_itsty + i) ^= j;
```

■ 割り込み関数の定義

LIST3-4.Cでは、(idotcsr) と名付けた関数を割り込み関数として定義しています。この例のように、割り込み関数は、interrupt 修飾子を付けて、その関数の型を void 型として宣言します。なお、この時のメモリモデルに制限はありません。

```
割込関数の定義
void interrupt test(void) {
```

また、LIST3-4.Cでは、割り込み関数を定義すると共に、カーソル表示をする関数と、ここで定義した関数を割り込みベクタテーブルへ登録する為の関数 (initcsr)、そして、割り込みベクタテーブルを復元するための関数 (termcsr) も定義してあります。

では、実際に LIST3-4.C で定義した関数を、先に組んだ関数と組み合わせてみましょう。メイン関数とプロジェクトファイルは次のようになります。

```
extern void initset (void);
extern void dotcsr(int);
extern void initcsr(void);
extern void termcsr(void):
extern void interrupt idotcsr(void);
int xmax, ymax;
void main()
作では、このよう
   grphinit();
   pointer_set();
initset();
   vramdotset(wwindow, 4, 0xff, 1, 2, 4, 8, xmax / 8, ymax, 0, 0);
   dispwaku();
 zoombitdt():
   initcsr();
   getch();
   termcsr();
   closegraph();
```

LIST3-5.PRJ

```
list2-0
list2-1
list3-0
list3-2
list3-4
list3-5
```

■ キーの入力を知るには(高速編)

キーボードからの入力を知る方法としては今までのように、TURBO Cの関数を使用するのが一般的であり、また、最も簡単な方法ですが、すこしでも高速な処理を要求される局面では、キーボード割り込みを直接利用するという方法をとります。特にリアルタイムゲーム等では、このような手法がしばしば用いられます。また、これから作ろうとしているパターンエディタのように、キーの入力が限定されている処理でも、やはり、直接利用するとツールの応答性が良くなります。

ここでは、割り込み処理の利用サンプルとしての目的も兼ねて、このキーボード割り込みを使ってみることにしましょう。

さて、キーボードからの割り込みは V-SYNC 割り込みとは異なり、等間隔ではなく、キーボード上のキーが押された、または、離された時に生じます。そして、この割り込みタイミングを利用して独自の関数へと制御を移すわけです。

この時のキーの情報は、ポートの $41_{\rm H}$ より得られます。なお、それぞれの キーとポート $41_{\rm H}$ より得られるデータとの対応は、表3-2を参照してください。

次のプログラムでは、使用するキーの入力に対応した処理をすると共に、 得られたキー情報は、外部変数の keydat へ格納して、現在のキー情報を他 の関数からも参照できるようにしてあります。

LIST3-6.Cでは、関数 keyset()をキーボード割り込みとして登録する 関数 initkey()と、割り込みベクタを元の状態へ復元する関数 termkey() を定義してあります。特に、関数 initkey()を起動した後は、関数 termkey ()を実行しなければ、キーの入力は限定されたままですから注意してくだ さい。では、プロジェクト LIST3-7.PRJ を実行してみましょう。

表3-2 キーコード表

押した時の出力データ	id T		0.				0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1
D7 D6	D5 D4	D3	D2	D1	DO		0) 1	2	3	4	5	6	7
	ortb	0	0	0	0	0	ESC	Qy	F _N	〈 ' ネ`	-		STOP	SHIFT
	iat e ceyei:	0	0	0	1	1	1,1	W	G _‡	`N°	/	NFER	COPY	CAPS
		0	0	1	0	2	2,7	E	H	/ 3.	7		f. 1	ħナ
		0	0	1	1	3	3 [#] 7	R	J	_ 	8		f. 2	GRPH
		0	1	0	0	4	4 7 7	T	K,	SPACE	9	7 42	f. 3	CTRL
		0	1	0	1	5	5 r	Yy	L	XFER	*	1	f. 4	eb de
		0	1	1	0	6	6 * *	U	; ,	ROLL U P	4	dota	f. 5	nieji
		0	1	1	1	7	7++	I _z	* : ₇	ROLL DOWN	5	(plov)	f. 6	o ble
		1	0	0	0	8	812	0,	١ ١	INS	6	- vo	f. 7	sk ke
		1	0	0	1	9	e ^E 6	P	Zy	DEL	**	0	f. 8	esi .
		1	0	1	0	A	0,7	@.	X	1	14	長の時	f. 9	-# -
		1	0	1	1	В	= - #	[. [{] r	Cy	+	2	1.) 1.)	f. 10	U 1011
		1	1	0	0	С	^	Ą	V _t	→	3	DIAN	efine	09 69
		1	1	0	1	D	¥_	A	В	+	9U_	EXAM EXAR	efine efine	6# 6#
		1	1	1	0	Е	BS	S	N _.	HOME CLR	0	EZEAM E	efine	bil hii
		1	1	1	1	F	TAB	D _{.y}	M _E	HELP	wọa	ENAM	enile	bğ
D7 D6	D5 D4	D3	D2	D1	D0	5 V	8	9	Α	В	С	D	Е	F
を		cas		15.5	edica KSE	SI	0 0 0 1	1 8 0 0 1	0 1 0 1	1 1 0 1	0 0 1 1	1 0 1 1	0 1 1 1	1 1 1

```
LIST3-6.C
#include <dos.h>
#include <bios98.h>
extern int
   ex,
   ey,
    xmin.
   xmax,
    vmin.
    ymax,
    space,
    counter,
    crson,
    dot adr,
  kaku adr,
    color,
    dot set;
extern void dotcrs (void);
void crsmov (void);
int keydat,
    shift key = 0,
    keysign = 0;
/* キーボード割り込み関数の定義 */
void interrupt keyset (void)
    #define MAKE RIGHT
                             0x48
                             0x46
    #define MAKE LEFT
    #define MAKE UP
                            0x43
    #define MAKE DOWN
                             0x4b
    #define MAKE UP RIGHT
                             0x44
    #define MAKE UP LEFT
                             0x42
    #define MAKE DOWN RIGHT 0x4c
    #define MAKE DOWN LEFT
                             0x4a
    #define MAKE ESC
                             0x00
    #define MAKE SPACE
                             0x34
                             0xb4
    #define BREAK SPACE
    #define MAKE SHIFT
                             0x70
     #define BREAK SHIFT
                             0xf0
```

```
int status_port = 0x43,
  comand_port = 0x43,
    data port = 0x41,
    eoi = 0;
  char command = 0x16,
  eoidata = 0x20;
                        - deard
  if(inportb(status_port) & 0x38); 1() vomezo
  else {
  outportb(comand_port, command);
  keydat = inportb(data port);
 if (keysign == 0) { switch(keydat) {
  case MAKE DOWN:
 if (++ey > ymax - 1) ey = ymin;
  11.0 .0 .8 .1 break; 11.18.13.70
          case MAKE_LEFT: _ abs_dob ) deadobmsxv
   if(-ex < xmin) ex = xmax - 1;
             break;
case MAKE RIGHT:
             if(++ex > xmax - 1) ex = xmin;
 break;
case MAKE_UP:
    if(--ey < ymin) ey = ymax - 1;
            break; as Hiss (ruink andha = Jabyas) hi
case MAKE DOWN_RIGHT:
    if (++ex > xmax-1) ex = xmin;
    if (++ey > ymax-1) ey = ymin;
    break;
    case MAKE DOWN_LEFT:
    if (--ex < xmin) ex = xmax - 1;
    void initk@y(if(++ey > ymax - 1) ey = ymin;
break;
case MAKE UP RIGHT:
if (++ex > xmax - 1) ex = xmin;
                        ey = ymax - 1; () 8x0300
             if(--ey < ymin)
         break;
         case MAKE UP LEFT:Z
                          ex = xmax - 1;
            if(--ex < xmin)
            if(--ey < ymin)
                         ey = ymax - 1;
break;
  case MAKE_SPACE:
            space = 1;
```

```
break;
            case BREAK SPACE: Ebx0 = Jaoq basmoo
              space = 0;
sinclude shipses we break;
            default :
               break:
        crsmov():
   if(space == 1) {
           crson = 0; (basamos , frog basago) dfroging
           dotcrs();
           if (ex & 1) vramdotset ( kaku adr + 80, color, 0x0e,
                             1, 2, 4, 8, 1, 3, 0, 0);
   else vramdotset ( kaku adr + 80, color, 0xe0,
                              1, 2, 4, 8, 1, 3, 0, 0);
   vramdotset ( dot adr, color, dot set,
  kaka ada, 1 - xama = x1, 2, 4, 8, 1, 1, 0, 0);
          crson = 1;
   dot set; dotcrs(); trees axam sead
   counter = 0;
wold or a now (void);
      if (keydat == MAKE SHIFT) shift key = 1;
if(keydat == BREAK SHIFT) shift key = 0;
 pablift key = 0.
   outportb(eoi, eoidata);
crson = 0;
  dotcrs();
   crson = 1; comban = xsidib- xanx < xs++) }}
  dotcrs(); MAYER STREET, We best (nimy > yo--) li
void interrupt (*keepvector9)();
void interrupt keyset(void);
/* キーボード割り込みの初期設定 */
void initkey(void)
```

```
keepvector9 = getvect(9);
setvect(9, keyset);
}

() diningro

/* キーボード割り込みを元にもどす */
void termkey(void)

(

KEY_INFO keyinf;
setvect(9, keepvector9);
keyinf.cmmd = 3;
bios98key(&keyinf);
}

() diningro

() dining
```

```
LIST3-7.C
/*
   キーボード割り込みのテスト
*/
#include <graphics.h>
extern void grphinit (void);
extern void pointer set (void);
extern void vramdotset (int, int, char, int, int, int,
                  int, int, int, int, int);
extern void zoombitdt (void);
extern void dispwaku(void);
extern void initset (void);
extern void dotcsr(int);
extern void initcsr(void);
extern void termcsr(void);
extern void initkey(void);
extern void termkey (void);
extern int xycset(int, int, int, int);
extern void interrupt idotcsr(void);
extern int
               色を自由に設定できるように、新たにドットの色を
   wwindow,
  keydat,
  space;
int xmax = 64,
   ymax = 64,
   xmin = 0,
   ymin = 0;
```

```
void main()
{
    grphinit();
    pointer_set();
    initset();
    vramdotset(wwindow, 4, 0xff, 1, 2, 4, 8, xmax / 8, ymax, 0, 0);
    dispwaku();
    zoombitdt();
    initcsr();
    initkey();
    keydat = 1;
    while(keydat != 0);
    termkey();
    termcsr();
    closegraph();
}
```

```
LIST3-7.PRJ

list2-0
list2-1
list3-0
list3-2
list3-2
list3-4
list3-6
list3-6
list3-7
list3-7
list3-7
list3-7
list3-7
list3-6
list3-7
list3-7
list3-7
list3-7
list3-7
list3-7
list3-7
list3-8
list3-7
list3-8
list3-7
l
```

実際にプロジェクト LIST3-7.PRJ を実行してみると、パターンエディタの基本構造がほとんどでき上がっていることがわかります。しかも、キーの操作性は合格点のはずです。

LIST3-7.PRJまで進んでくると、だいぶパターンエディタらしくなってきましたが、まだドットの色が固定されたままです。そこで、ドットの色を自由に設定できるように、新たにドットの色を選択する為の機能を付加することにしましょう。

```
extern void vramdotset (int, int, char, int, int, int,
      int, int, int, int);
extern int color, keydat;
void ccsrmov(int, int);
int kpcolor = 2; ology 是第一个意识解的本族大步可以各个數語多面包
/* カラー用カーソルの右への移動 */
void rcolor (void)
 を備しておいて、カーソルの移動によって指定することにしました。
if(++color > 15) color = 0;
  ccsrmov(color, kpcolor);
  kpcolor = color; HURRENTE ON V - ts. () voormoo MEELEES
/* カラー用カーソルの左への移動 */
void lcolor (void)
また、ここで注意したいのは、カーソル表示用の関数でStratovillの製
if(--color < 0) color = 15;
  ccsrmov(color, kpcolor);
kpcolor = color;
}
/* カラー用カーソルの移動 */
void ccsrmov(int cx, int kx)
  xorvramdt(CCURSOR + kx, 0xfe, 7);
 xorvramdt (CCURSOR + cx, 0xfe, 7);
  vramdotset (CSIGNE, cx, 0xff, 1, 2, 4, 8, 16, 7, 0, 0);
  keydat = 1; ayday 0.8-ET211
/* カラー選択用テーブルの初期セット */
void initcolor(void)
  int i, j;
  for(i = 0; i < 16; ++i) {
     j = CTABLE + i;
     vramdotset(j, i, 0xfe, 1, 2, 4, 8, 1, 7, 0, 0);
  xorvramdt (CCURSOR + color, 0xfe, 7);
  vramdotset(CSIGNE, color, 0xff, 1, 2, 4, 8, 16, 7, 0, 0);
```

さて、ドットの色を指定する場合、どの様な方法を使うかが問題になります。たとえば、色と番号を対応させて、番号によって選択するという方法などが考えられます。しかし、色の数が16色にもなると、色と番号との対応を記憶するだけでも大変な作業になってきます。

そこで、色を確認しながら選択できるように、ここでは色のテーブルを 準備しておいて、カーソルの移動によって指定することにしました。

LIST3-8.C では、色のテーブル表示用に関数 initcolor()、色用のカーソル表示に関数 ccsrmov()、カーソルの右方向移動用に関数 rcolor()、そして、カーソルの左方向移動用に関数 lcolor()の4つの関数を用意してあります。

また、ここで注意したいのは、カーソル表示用の関数 ccsrmov()の実行後、キーボード割り込みによって得られた変数 (keydat) の情報は必要がなくなりますから、プログラムに影響を与えない数値 (keydat =1) に書き換えているということです。このような細かい配慮も忘れないようにしてください。

では、実際に実行してみましょう。メインプログラムとプロジェクトファイルは次のようになります。

LIST3-9.C

#include <graphics.h>

#include <dos.h>

#include <bios98.h>

#define MAKE RIGHT 0x39

#define MAKE LEFT 0x38

extern void grphinit (void);

extern void pointer set (void);

extern void zoombitdt (void);

extern void dispwaku (void);

extern void initset (void);

extern void initcsr(void);

extern void termcsr(void);

```
extern void initkey(void); [ [ [ ] ] ] [ ] [ ]
extern void termkey (void);
extern void rcolor (void);
extern void lcolor(void);
extern void initcolor(void);
extern int
  wwindow,
  color,
  keydat;
int xmax = 64.
  ymax = 64,
  xmin = 0.
  がし、これだけでは、使えるツールとは領世辞にもいえます。O miny
woid main() 公共的社会公共的人会对加入会计和社会会会
改の LIST3-10.C では編集エリアを一定のパターンで埋める関数が
  grphinit();
  pointer_set();
  initset();
  dispwaku();
  zoombitdt();
  initcsr();
  initkey();
  initcolor();
  keydat = 1; Jakagani ani ando ani ani ani leetobeery blov eretus
  while (keydat != 0) 14 , 3at , 3al , 3al , 3al
   switch (keydat) {
   case MAKE RIGHT:
   rcolor();
          break;
        case MAKE LEFT:
           lcolor();
           break:
        default :
  こでのボインbreak: カーソルの取り扱いです。。例如は
  は、カーソルの位置を示すドットが点滅してい
  termkey();
  termcsr(); de dal adaba and aclos dat assets ant) talaglis blov
   closegraph();
  のドットの点滅は、割り込みによって処理されています。よどmtトは
```

LIST3-9.PRJ					
14-+0 1	は、色と養考を対応させて、養考によりで運搬するという。 (blov) rotopr blov nzabze				
list3-0					
list3-2					
list3-4					
list3-6					
list3-8					
list3-9					

プロジェクト LIST3-9.PRJ では、任意のキャラクタパターンを編集できるようになったわけです。いわば木の幹ができたといえるでしょう。しかし、これだけでは、使えるツールとは御世辞にもいえません。すなわち、使える状態にするには、さらに、枝や葉を付加しなければならないのです。

次のLIST3-10.Cでは編集エリアを一定のパターンで埋める関数を組んでいくことにしましょう。

LIST3-10.C

void allpaint (int addres, int color, char data, int shift)

int i, j, k, 1;

```
dotcrsoff();
 1 = xmax / 8;
  for(j = 0; j < ymax; ++j) { 3 3 3 4 5 4 5 () Hoerotob . (
  for(i = 0; i < 1; ++i, ++addres) {
  vramdotset(addres, color, data, 1, 2, 4, 8, 1, 1, 0, 0);
  baint()を利用して、編集エリアを5種類のパターンで埋めるように
    data = _rotl(data, shift);
    addres += k:
  ットの色は、現在選択されているドットの色ということにい訳して
  zoombitdt();
  dotcrson();
/* カーソルをオンとする */
void doterson (void)
  crsinf = 1;
  crson = 1;
  dotcrs();
  counter = 0;
  keysign = 0;
}
/* カーソルをオフとする */
void dotcrsoff (void)
  crsinf = 0;
  crson = 0;
 dotcrs();
1
```

ここでのポイントは、カーソルの取り扱いです。実は、左上の編集エリ アには、カーソルの位置を示すドットが点滅していたのです。せっかく凝っ て作ったのですから、まったく気付かなかったという人はいないでしょう tan

このドットの点滅は、割り込みによって処理されています。ドットは、 XOR によって表示、消去を繰り返しています。

ですから、このドットの点滅を一時的にストップさせなければならないのです。このドットの点滅を管理しているのが LIST3-10.C の dotcrson (), dotcrsoff()という関数で、単純にコールするだけとしました。

編集エリアをパターンで埋める関数は allpaint()となります。この allpaint()を利用して、編集エリアを5種類のパターンで埋めるようにしたのが次のプロジェクト LIST3-11.PRJ です。なお、編集エリアを埋める時のドットの色は、現在選択されているドットの色ということにしました。

```
LIST3-11 C
#include <graphics.h>
#define MAKE RIGHT 0x39
#define MAKE LEFT
                0x38
#define MAKE F1
                0x62
#define MAKE F2
                0x63
#define MAKE F3
                0x64
#define MAKE F4
                0x65
#define MAKE F5
                0x66
extern void grphinit (void);
extern void pointer set (void);
extern void zoombitdt (void);
extern void dispwaku(void);
extern void initset (void);
extern void dotcsr(int);
extern void initcsr(void);
extern void termcsr (void);
extern void initkey (void);
extern void termkey (void);
extern void rcolor (void);
extern void lcolor (void);
extern void initcolor(void);
extern void allpaint (int, int, char, int);
extern int was a washe washe washe a water
wwindow, the eddress, the color, that data, inc shift)
   color,
keydat, TEADINEEL TO LEAKE OF LINE NOT WHO I
```

```
space:
int xmax = 64,
  ymax = 64,
  xmin = 0,
  ymin = 0;
void main()
  grphinit();
  pointer set();
  initset();
  dispwaku();
  zoombitdt();
  initcsr();
  initkey();
  initcolor();
  keydat = 1;
  while (keydat != 0) {
     switch (keydat) {
       case MAKE RIGHT:
          rcolor(); ______ 199 11-87211 4 4 4 6 0 7 3 %
          break:
       case MAKE LEFT:
          lcolor(); II a de Har V V a Mar Jac 103 av 53
          break:
       case MAKE F1:
         allpaint(wwindow, color, 0xff, 0);
          break;
       case MAKE F2:
          allpaint (wwindow, color, 0xaa, 1);
          break:
       case MAKE F3:
         allpaint (wwindow, color, 0x55, 1);
          break;
       case MAKE F4:
          allpaint (wwindow, color, 0xaa, 0);
          break;
       case MAKE F5:
  allpaint (wwindow, color, 0x55, 0);
  break;
  default :
```

```
break;

}

termkey();

termcsr();

closegraph();
```

LIST3-1	
list2-0	(f) Jes_salajog
list2-1	
list3-0	
list3-2	
list3-4	
list3-6	
list3-8	
list3-10	
list3-11	

さて、プロジェクト LIST3-11.PRJ のキー操作は、ファンクションキーの f1~f5となっています。それぞれ、ドットカラーを変更して試してみてください。このように、編集エリアを埋める関数が付加されただけでもかなり使い勝手がよくなったはずです。ファンクション機能をさらに増やして、色々なアイデアを付加してみるのも面白いかもしれません。

ところで、実際にパターンを編集してみると、パターンはひとつだけとは限りません。複数のパターンを編集する場合がほとんどです。しかも、以前に編集したパターンを一部分だけ変更したい場合も多々あります。そこで、編集し終ったパターンはどこかに退避させて、必要な時に編集エリアへと再び呼び出せるような構造にしておきたいものです。

この退避するエリアとしては、当然のことながらメモリがあげられますが、編集した絵を把握するという観点から、空いている画面(G.V.RAM)を利用するという方法が有利といえます。

LIST3-12 C

```
extern int
   keydat,
   xmax.
   vmax,
   wwindow,
   shift key;
extern char far *c_blue;
extern char far *c red;
extern char far *c green;
extern char far *c itstv;
extern void dotcrson (void);
extern void dotcrsoff (void);
extern void zoombitdt (void);
void abchange(int, int);
void boxcursol(int);
void boxmov(void);
int box inf,
   box adr = 36,
   fukugen = 10,
   selectw = 19,
   mwindow = 36,
          = 0,
   bx
           = 0:
   by
編集パターンを指定エリアへ退避、または退避
エリアから編集エリアへパターンを読み込む
*/
void ptn copy (void)
   int i, j, k, w;
   #define MAKE READ
                       0x13
   #define MAKE WRITE
                       0x11
   #define MAKE FUKUGEN 0x20
   #define MAKE ZOOM
                       0x29
   #define MAKE RIGHT
                       0x48
   #define MAKE LEFT
                       0x46
   #define MAKE UP
                       0x43
```

```
0x4b 01-8T8L
#define MAKE DOWN
dotcrsoff():
box inf = 1;
w = 1;
boxcursol(box adr);
while (shift key == 1) {
    switch(keydat) {
        case MAKE READ:
        case MAKE ZOOM:
            boxcursol(box_adr);
            abchange (fukugen, wwindow);
            abchange (wwindow, box adr);
            boxcursol(box adr);
            if (keydat == MAKE ZOOM) zoombitdt();
            w = 0;
            keydat = 1;
            break:
        case MAKE WRITE:
            boxcursol(box adr);
            abchange (fukugen, box adr);
            abchange (box adr, wwindow);
            boxcursol (box adr);
            w = 1;
            kevdat = 1;
            break;
        case MAKE FUKUGEN:
            boxcursol(box adr);
            if(w == 0)
                            abchange (wwindow, fukugen);
                            abchange (box adr, fukugen);
            boxcursol(box adr);
            keydat = 1;
            break;
        case MAKE RIGHT:
            if(++bx > 44 - xmax / 8) bx = 0;
            boxmov();
            break;
        case MAKE LEFT:
            if(--bx < 0) bx = 44
                                   xmax /
            boxmov();
            break;
        case MAKE DOWN:
```

```
if(++by > 50 - ymax / 8) by = 0;
           boxmov():
           break:
        case MAKE UP:
          if(--by < 0) by = 50 - ymax / 8;
           break:
 boxcursol(box adr);
  dotcrson();
/* ボックス型カーソルの移動 */
void boxmov (void)
  boxcursol(box adr);
  box adr = by * 8 * 80 + bx + mwindow;
  abchange (selectw, box adr);
  boxcursol (box adr);
  keydat=1;
extern void xorvramdt (int, int, int);
/* ボックス型カーソルの表示 */
void boxcursol(int k)
  int i, j;
  if(box_inf == 1) {
     j = (ymax - 1) * 80;
     for(i = 0; i < xmax / 8; ++i, ++j) {
        xorvramdt (k + i, 0xff, 1);
        xorvramdt(k + j, 0xff, 1);
 j = xmax / 8 + k + 79;
     k += 80;
 xorvramdt(k, 0x80, ymax - 2);
 xorvramdt(j, 0x01, ymax - 2);
  else box inf = 1;
```

```
/* bからaへのパターンのコピー */ 18 * xamy - 08 * yd++) *1
void abchange(int a, int b)
   int i, j, k;
   for (j = 0, k = 0; j < ymax; ++j)  { (0 > yd--) }
   for(i = 0; i < xmax / 8; ++i, ++k) { } vormed
   *(c blue + a + k) = *(c blue + b + k);
      *(c \text{ red } + a + k) = *(c \text{ red } + b + k);
           *(c green + a + k) = *(c green + b + k);
           *(c itsty + a + k) = *(c_itsty + b + k); rod) loss of
       k += 80 - xmax / 8;
```

■ G.V.RAM 上のパターンのコピー

ここで、G.V.RAM上のパターンをコピーする手順を考えてみます。まず、画面上にある編集したパターンAを単純にA'の位置へとコピーするわけですから、それぞれの、基準となるアドレスはそれぞれのパターン左上のアドレスを求めればよさそうです。となりのアドレスは単純に+1すればよく、1ライン下のアドレスはG.V.RAMの構成から+80すればよいからです。ここで問題となるのは、編集エリアのアドレスは固定ですから簡単に求まりますが、移動すべきA'のアドレスをどのように決定するかです。

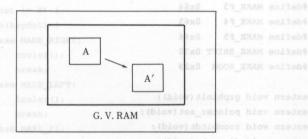


図3-1 AをA'ヘコピーする

表3-3 BOX カーソルの操作方法

[chift] + [2] | how カーソルも下去向へ移動する

	DOXカーブルを下方向へ移動する
[shift] + [4]	box カーソルを左方向へ移動する
[shift] + [6]	box カーソルを右方向へ移動する
[shift]+ [8]	box カーソルを上方向へ移動する
[shift]+ [W]	編集エリアのパターンをboxカーソルで示されるエリア
	ヘコピーする
[shift]+ [R]	boxカーソル位置のパターンを編集エリアにコピーする
[shift]+ [F]	直前のコピー動作によってコピーしたパターンを元に
CARR	戻す。wobnisse
[shift] + [Z]	boxカーソルのパターンを編集エリアへコピーするとと
	もに拡大表示する

そこで、LIST3-12.Cでは、感覚的にわかりやすい box 型のカーソルを 移動することによってコンピュータに計算させることにしました。この時 のキーの操作方法は表3-3の通りとなります。

なお、box 型のカーソルを表示した時には、編集用の点滅カーソルの表示はカットしてあります。では、プロジェクト LIST3-13.PRJ を起動してみましょう。

```
LIST3-13.C
#include <graphics.h>
#define MAKE RIGHT 0x39
#define MAKE LEFT 0x38
#define MAKE F1
              0×62
#define MAKE F2
              0x63
#define MAKE F3
              0x64
#define MAKE F4
              0x65
#define MAKE F5
              0x66
#define MAKE SHIFT 0x70
#define MAKE ZOOM 0x29
extern void grphinit (void);
extern void pointer_set(void);
extern void zoombitdt (void);
extern void dispwaku (void);
extern void initset (void);
extern void dotcsr(int);
extern void initcsr(void);
extern void termcsr(void);
extern void initkey(void);
extern void termkey (void);
extern void rcolor(void);
extern void lcolor(void);
extern void initcolor(void);
extern void allpaint (int, int, char, int);
extern void ptn copy(void);
extern int
   wwindow,
 Color, SEATURE & C-ENOUGE - Axod
   keydat,
```

```
space;
int xmax = 64,
  ymax = 64,
xmin = 0,
  vmin = 0;
void main()
Textern void vrandotset (int. int. oner tet) one int () 25000391
grphinit(); int, int, int, int, let ye () Mgarpezolo
  pointer set();
initset(); dispwaku();
zoombitdt();
  initcsr();
 initkey();
  initcolor(); and all objects and all of
  kevdat = 1;
  while (keydat != 0) {
  switch(keydat) {
  case MAKE RIGHT:
  rcolor();
  for(4 = 1) break;
    case MAKE LEFT:
       lcolor();
         break;
case MAKE F1:
   allpaint(wwindow, color, 0xff, 0);
         break;
    case MAKE F2:
        allpaint(wwindow, color, 0xaa, 1);
         break:
case MAKE F3:
         allpaint (wwindow, color, 0x55, 1);
  break;
  case MAKE F4:
         allpaint (wwindow, color, 0xaa, 0);
        break;
  case MAKE F5:
        allpaint(wwindow, color, 0x55, 0);
                     extern char far *c green;
  break;
  case MAKE_SHIFT:
         ptn copy();
         break;
```

```
case MAKE_ZOOM:
    zoombitdt();
    break;
    default :
        break;
}
termkey();
termcsr();
closegraph();
}
```

```
LIST3-13.PRJ

list2-0
list2-1
list3-0
list3-2
list3-4
list3-6
list3-8
list3-10
list3-12
list3-12
```

さて、これでパターンエディタの編集機能は最低限、そろったわけです。 でも、せっかくここまで作ったのですから、もう少し編集機能を充実させ たいものです。そこで、編集エリアの回転移動、平行移動、上下反転機能 を付加することにしました。なお、左右の反転は、回転移動と上下反転に よって実現することができますから、ここでは機能としては省いてありま す。

```
extern char far *c_blue;
extern char far *c_red;
extern char far *c_green;
extern char far *c_itsty;
extern int
xmax,
```

```
ymax,
  wwindow, see tests a sol (wobnies , neguxal) egasdods
  fukugen; - 4, 14 - 10 - 8, 04 - 05 + 07 to 10 mm 8 \ manus = 0
extern void dotcrson(void); (08 ** 01 ,08 ** 00 ,14*
extern void dotcrsoff(void);
extern void zoombitdt (void);
extern void abchange(int, int);
extern void vramdotset (int, int, char, int, int, int,
(wo++ wi++ (d++) int, int, int, int, int); 0 = 11001
void rolling(void) was ( ow. 0x80, 0x80(wkeys, 48800 0) * * 0
  int b, r, g, l, i, j, j2, k, o, o2, m, k2;
  unsigned char n, n2, nb, nr, ng, n1; (wo bear o)
  dotcrsoff();
  abchange (fukugen, wwindow); 08x0 [1x0 wo ] 1xx 1cb xxx
  k = fukugen + xmax / 80 - 1;
  o2 = wwindow;
  for(j = 0; j < ymax / 8; ++j, ++o2) {
     for (j2 = 0, n2 = 0x80; j2 < 8; ++j2, n2 >>= 1, k += 80) {
        for (i = 0, n = 0x1, o = o2, k2 = k; d = fod
  i < xmax / 8; ++i, --k2, o += 8 * 80) {
  nb = *(c blue + k2);
                      + k2);
           nr = *(c red
   ng = *(c green + k2);
           nl = *(c itsty + k2);
   overses vramdotset(o, n, n2, nb, nr, ng, nl, 1, 8, 1, 0);
   1.5 (k == 1)
     1-88:
   dotcrson(); (igst popt and dost i .p. a .d and bengianu
/* 編集パターンを右方向へ1ドット移動する */
void dotshiftr(void)
   int i, e, i0, iw, j, o0, ow; ) (08 =+ 02 ,08 =+ 00 ,1++
   unsigned char b, r, g, l, kpb, kpr, kpg, kpl;
```

```
dotcrsoff();
   abchange (fukugen, wwindow);
      = xmax / 8 - 1;
   for(j = 0, i0 = fukugen, o0 = wwindow; j < ymax;
      ++j, o0 += 80, i0 += 80) { (blow) neeradob blow meadas
      kpb = *(c blue + i0 + e);
      kpr = *(c_red + i0 + e); (blow) shildmoon blow mrates
   kpg = *(c green + i0 + e);
   kpl = *(c_itsty + i0 + e); Jak Jak ) Jesjohnstv blov mejks
   for (i = 0, iw = i0, ow = o0; i < xmax / 8; ++i, ++iw, ++ow) {
         b = *(c blue + iw);
         r = *(c red + iw);
         g = *(c green + iw);
         1 = *(c itsty + iw);
         *(c blue + ow) = b >> 1;
 *(c_red + ow) = r >> 1; 11 , dn , 21 , n rado benplano
        *(c green + ow) = g >> 1;
        *(c itsty + ow) = 1 >> 1;
 vramdotset( ow, 0x1, 0x80, kpb, kpr, opposite opening
                  kpg, kpl, 1, 1, 0, 0);
 kpb = b;
         kpr = r; ) (So++ (++ 18 \ xsmy > ( 10 = () xo)
         kpg = q; Sn ,Sr++ :8 > Sr :08x0 = Sn .0 = Sr:202
         kpl = 1; 1x = 2x .50 = 0 .fx0 = n .0 = 1) rol
まて、これでバクーンエディク、犯罪集機能は最低限率もあったわけです。
  zoombitdt();
  dotcrson();
/* 編集パターンを左方向へ1ドット移動する */
void dotshiftl(void)
  int i, e, i0, iw, j, o0, ow;
  unsigned char b, r, g, l, kpb, kpr, kpg, kpl;
  dotcrsoff();
     = xmax / 8 - 1;
  for(j = 0, i0 = fukugen, o0 = wwindow; j < ymax;</pre>
      ++j, 00 += 80, i0 += 80) { wo .00 ; wh .01 .0 i drik
     kpb = *(c blue + i0);
     kpr = *(c red + i0);
```

```
kpg = *(c green + i0);
     for (i = 0, iw = i0 + e, ow = o0 + e; i < xmax / 8;
  b = *(c_blue + iw); all o)* * (1 + 1 + visil o)*
        r = *(c red + iw);
        q = *(c green + iw);
        l = *(c itsty + iw);
        *(c blue + ow) = b << 1;
        *(c red + ow) = r << 1;
        *(c green + ow) = g << 1;
        *(c_itsty + ow) = 1 << 1;
        vramdotset ( ow, 0x80, 0x01, kpb, kpr,
             kpg, kpl, 1, 1, 0, 0); b a d d dal
        kpb = b;
       kpr = r;
        kpg = g;
        kpl = 1;
zoombitdt();
  dotcrson(); ____ | ___ (b++ ,b++ ,b++ ,b++ ,b \ xamx > 1 (0 = 1) xol
void dotshiftud(int k)
  int i, j, l, m;
  dotcrsoff(); (b + x + assap a)^* = (a + x + assap a)^*
  if(k == 0) {
     j=80;
     1 = wwindow + ymax * 80 - 80;
     m = fukugen;
  else {
     j=-80;
     l = wwindow;
     m = fukugen + ymax * 80 - 80;
  abchange (fukugen, wwindow);
  abchange (wwindow, fukugen + j);
```

```
for (i = 0; i < xmax / 8; ++i) { (01 + means s) = post
 *(c blue + l + i) = *(c blue + m + i);
 *(c red + 1 + i) = *(c red + m + i);
 *(c_green + 1 + i) = *(c green + m + i);
   *(c_{itsty} + 1 + i) = *(c_{itsty} + m + i);
  dotcrson();
void udhanten (void)
  int i, j, k, u, d; 100 .001 .1 .101 .pg/
  char m; to blok a swheels be like id a dok
 dotcrsoff();
 u = 0; Me issee + set = 1 >> 1;
 d = (ymax - 1) * 80;
  k = wwindow;
 for(j = 0; j < ymax / 2; ++j) {
    for(i = 0; i < xmax / 8; ++i, ++u, ++d) {
      m = *(c blue + k + u);
      *(c_blue + k + u) = *(c_blue + k + d);
   m = *(c red + k + u); (stant) bushingsob blow
 *(c red + k + u) = *(c red + k + d);
 *(c red + k + d) = m;
 m = *(c_green + k + u);
     *(c_green + k + u) = *(c_green + k + d); 1(1)10010100
 *(c green + k + d) = m;
 m = *(c itsty + j + u);
      *(c itsty + k + u) = *(c itsty + k + d);
 *(c_{itsty} + k + d) = m;
 unalled that by my g. 1, kpb, kpc, kpg, kpl;
    u += 80 - xmax / 8;
 d -= 80 + xmax / 8;
 zoombitdt();
 dotcrson();
```

このプログラムでは、ビット操作をしていますから、コンピュータの扱っている数値が、2進数であったということを忘れてしまうと、ほとんど、プログラムとしては分かりにくいものとなってしまいます。では、さらに、次の LIST3-15.C に進んでください。

```
LIST3-15.C
#include <graphics.h>
#define MAKE CR
                     0x39
#define MAKE CL
                     0x38
#define MAKE F1
                     0x62
#define MAKE F2
                     0x63
#define MAKE F3
                     0x64
#define MAKE F4
                     0x65
#define MAKE F5
                     0x66
#define MAKE SHIFT
                     0x70
#define MAKE ZOOM
                     0x29
#define MAKE ROLL
                     0x13
#define MAKE RIGHT
                     0x3c
#define MAKE LEFT
                     0x3b
#define MAKE UP
                     0x3a
#define MAKE DOWN
                     0x3d
#define MAKE HANTEN 0x16
extern void grphinit (void);
extern void pointer set (void);
extern void zoombitdt (void);
extern void dispwaku (void);
extern void initset (void);
extern void dotcsr(int);
extern void initcsr(void);
extern void termcsr (void);
extern void initkey(void);
extern void termkey (void);
extern void rcolor (void);
extern void lcolor(void);
extern void initcolor (void);
extern void allpaint (int, int, char, int);
extern void ptn copy(void);
extern void rolling (void);
```

```
extern void dotshiftr(void);
extern void dotshiftl(void); gig klas / 155 day milks / 16 milks / 16
extern void dotshiftud(int);
extern void udhanten(void);
    COLISTS-15.Cに選択でくれまままた。つき - (1 + 1 + ytest o)*-
extern int
  wwindow,
  color,
  keydat,
  space;
int xmax = 64,
  ymax = 64,
  xmin = 0,
  vmin = 0:
void main()
  grphinit();
  pointer set();
  initset();
  dispwaku();
  zoombitdt();
  initcsr(); have a key as we blue
  initkey();
  initcolor();
  keydat = 1; yad a keydat a keydat a
  while (keydat != 0) {
     switch(keydat) {
        case MAKE_CR:
        rcolor();
        break;
        case MAKE CL:
          lcolor();
          break;
      case MAKE F1:
     allpaint(wwindow, color, 0xff, 0);
          break:
    case MAKE F2:
          allpaint (wwindow, color,
                            0xaa, 1);
          break;
        case MAKE F3:
```

```
allpaint (wwindow, color, 0x55, 1);
                                     break:
                            case MAKE F4:
                                     allpaint (wwindow, color, 0xaa, 0);
                                   break; was a superior to the s
                            case MAKE F5:
        allpaint (wwindow, color, 0x55, 0);
                          break;
                            case MAKE SHIFT:
                           ptn copy();
          case MAKE ZOOM:
zoombitdt();
                                     break; A LOT- 9 4 MEO-4 E U LOT-
                            case MAKE ROLL:
     rolling();
                                     break;
     case MAKE_RIGHT:
                                     dotshiftr();
                                     break;
     case MAKE_LEFT:
                                     dotshiftl();
                            case MAKE_UP:
                                     break:
                                     dotshiftud(0);
                                      break;
                             case MAKE DOWN:
             dotshiftud(1);
                                      break;
                             case MAKE_HANTEN:
           udhanten();
                                      break;
                            break;
          termkey();
          termcsr();
          closegraph();
```

```
LIST3-15.PRJ
list2-0
list2-1
list3-0
list3-2
list3-4
list3-6
list3-8
list3-10
list3-12
list3-14
list3-15
```

■ グラフィック画面のセーブ/ロード

パターンエディタもようやく仕上げとなります。パターンエディタとしては、プロジェクト LIST3-15.PRJ で終わりですが、作成したパターンをディスクへセーブしたりロードしたりできなければ、ツールとしては不完全です。ここでは、ファイル管理について話を進めていきます。

G.V.RAM上にある編集したデータを単純にディスクへファイルとして保存したり、あるいは、ディスクにあるファイルをG.V.RAMへロードするわけですが、G.V.RAM上のデータも他のメモリ上のデータとなんら変わる所はありません。ですから、操作手順としては次のようになります。

手順2:ファイルをセーブ/ロードする

手順3:ファイルをクローズする

この時、ロードする場合であればロード先のメモリが G.V.RAM であり、セーブする場合であれば、データの存在するメモリが G.V.RAM であるというだけです。

なお、ファイルには管理するためのポインタであるファイルポインタがあり、すべてのファイルはこのファイルポインタを使って管理しています。 LIST3-16.Cでは関数 fopen()でファイルをオープンし、関数 fgetc(), fputc()でディスクへの読み書きを、関数 fclose()でファイルのクローズを実現しています。

LIST3-16.C

#include <dir.h>

#include <io.h>

#include <stdio.h>

#include <comio.h>

#include <bios98.h>

```
#define P SHIFT 1
 extern int mwindow;
 extern char far *c blue;
 extern char far *c green;
 extern char far *c itsty;
                                           ネスクヘセーブしたりロードしたりできなければ、
 void fmes(int, char []);
 void file disp(char []);
 int getname(char []);
/* グラフィック画面データをディスクへセーブする */
 gsave (void) {
         char fname[64];
 TIFILE *sfp; ①X117J3图手的键 , act 5 , A 9 1 0 8 11 图 3 ct
         char c:
         int i, j, k, l, fsin;
         fsin = 0;
         while(1) {
                 i = 3:
                  while(i) {
printf("\forall name\forall name\fora
i = getname(fname);
                         if(i == 1) {
                                   fsin = 2;
なお、ファイルには管理するためのポインタで;Sead ティルポインタル
if(i == 2) file disp(fname);
 () oteo) 機関else i=0; 一大タイトマでつ() negol 機関はつうるにと
                  if(fsin == 2) break;
                   if(access(fname, 0) == 0) {
                           printf( "¥n%s is allready exist ¥ncontinue (Y-N) ? ",
                                            fname );
                           while(1) {
                                    c = getch();
                                 if(c == 'n' || c == 'N') {
                                        fsin = 2;
                                      break;
                                    }
```

```
if(c == 'Y' || c == 'y') {
              printf("Y");
              break;
     priotf("Yningut file name = %aVn", name); startd
   if (fsin == 2) break;
     else printf("save file name = %s ", fname);
  sfp = fopen(fname, "wb");
 if (NULL == sfp) {
       fsin = 3;
   break;
  strolt resik fronk:
     for(i = 0, 1 = mwindow; i < 400; ++i, 1+=80) { do) emanage and
  for (j = 0, k = 1; j < (80 - mwindow); ++j, ++k)
     fsin = 1;
        if(fputc(*(c blue + k), sfp) == EOF) break;
        if(fputc(*(c red + k), sfp) == EOF) break;
        if(fputc(*(c green + k), sfp) == EOF) break;
        if(fputc(*(c_itsty + k), sfp) == EOF) break;
        if(fsin == 1) break; (FdV F) Adming
break;
  bhar frammi64).
  fmes(fsin, fname);
  return(fsin);
/* エラーメッセージの表示 */
void fmes(int fsin, char fname[])
  clrscr();
  gotoxy(1, 5);
  switch(fsin) {
     case 0:
        printf("* normal end *");
        break; in = 2 (%sand (0 == [1]sman));
     case 1: behalff' am [leman | | + + + mow (leman) 11
        printf("¥7* abnormal end *");
```

```
break;
     case 2:
     printf("* cansel *");
     break:
     case 3:
     printf("¥7 cannot open file : %s", fname);
     break;
     default: (mant, " sa + man silt evas") training sale
break:
vold, file_disp(chas file ___ ALUM) li
int getname (char [])
/* ファイル名の入力 */
int getname(char name[])
  int i, j;
   j = 0; send (%08 ** (gts ,(x + suid o)*)osuql) 11
   for (i = 0; i < 40; ++i) { (4 + bes of *) of max) 1.
     while (1) { = (gis (1 + neers 5) *155ug2) 11
   name[i] = getche();
   if(name[i] == 0x8) {
      if(i > 0) --i;
      printf(" \b");
         printf("YnWhinput file nameWn");
         else break; (gta) esolot
      1 12(1 -- 1) (
      if(name[i] == 0x1b) {
         printf(" ");
         i = 1;
         break;
      ) slas isli
      if(name[i] == 0xd) {
      printf("Yn"); ([]emsnl xsdo ,niel snl) semi blow
      name[i] = 0;
         break; "Water all ready exist Encontinue. (Thibetibe
   if(j != 1) {
      for (i = 0; i < 40; ++i) { the famous **) istaling
         if(name[i] == 0) break;
         if(name[i] == '*' || name[i] == '?') {
            j = 2;
```

```
break;
   7741
    printf("\forall ninput file name = \forall s\forall n', name);
  return(j);
使うだけですから問題ありませんが、12gTh 指名 特性的oなるのは
/* ファイルの表示 */
void file_disp(char name[])
  struct ffblk ffblk;
   if (findfirst (name, &ffblk, 0) == 0) {
 printf("Yn%s", ffblk.ff_name);
        if(findnext(&ffblk) != 0) break;
printf("\forall t\forall s\sigma", ffblk.ff_name);
        if(findnext(&ffblk) !=0) break;
德翁]亦中中人上了(以徐时代 郑明 美国等等的时间的最大的科学——下海的汉克东
/* グラフィック画面データをディスクからロードする */
gload(void) {
  char fname[64];
  FILE *sfp; (Maserd (THIHE T & (Inkyesta) yest@cold) 11
   int c, i, j, k, l, s, done, fsin;
   struct ffblk ffblk;
  KEY INFO keyinf;
   fsin = 0;
   while(1) {
   1 = 3: (VOIG)
     while(i) {
        printf("\forall n \text{ Yn \forall n in put file name \forall n");
        i = getname(fname);
        if(i == 1) {
           fsin = 2; t, int, int, int, int );
           break;
```

```
if(i == 2) file disp(fname);
   else i = 0:
   I server the cannot see
   if (fsin == 2) break; ala = omen sill sugalay ) timing
   printf("load file name = %s ", fname);
   sfp = fopen(fname, "rb");
   if (NULL == sfp) {
   fsin = 3;
     break:
   kevinf.cmmd = 3;
   bios98key(&keyinf);
for(i = 0, 1 = mwindow; i < 400; ++i, 1+=80) {
for (i = 0, k = 1; i < (80 - mwindow); ++i, ++k) {
         fsin = 1: 1 (0 == 10 .xidlla .sman) faxllball) 11
        if((c = fgetc(sfp)) == EOF) break;
*(c blue + k) = c;
if((c = fgetc(sfp)) == EOF) break;
   *(c_red + k)=c;
       if((c = fgetc(sfp)) == EOF) break;
      *(c \text{ green} + k) = c;
         if((c = fgetc(sfp)) == EOF) break;
         *(c itsty + k) = c;
         fsin = 0;
      /* グラフネック展面データをディスクからロードする *广 titabtd walls
      if(fsin == 1) break;
   keyinf.cmmd = 2;
      if (bios98kev(&keyinf) & P SHIFT) break;
   fclose(sfp);
   break;
keyinf.cmmd = 3;
bios98key(&keyinf);
fmes(fsin, fname);
return(fsin);
```

ファイル名の入力

#include <dir.h>
#include <io.h>

ex,

さて、ファイルのセーブ/ロードの手順は、それぞれ TURBO C の関数を使うだけですから問題ありませんが、LIST3-16.C で問題となるのは、ファイル名の入力方法でしょう。人間にはミスがつきものですし、人間は、わがままですから入力の途中で中断したい場合もあるからです。

そこで、LIST3-16.C では比較的自由度を持たせたファイル名入力専門の関数 getname を用意ました。特徴は、一文字ずつキー情報を取り込んで引数の配列 name[]へ格納していることです。というのも、[ESC] キーが入力された場合には速やかに処理を中断するようにしたいからです。また、配列ですから[BS] キーが入力された時にはインデックスを単純に一1することによって以前のキー入力値を取り消すことができるというメリットもあります。さらに、ワイルドカードの入力もチェックできるように、リターン情報をセットしています。なお、LIST3-17.C はキーボード割り込みをコントロールするためのものです。

LIST3-17.C

```
void sideclr(int);
/* ファイル操作初期設定 */
void fstart (void)
使うだけですから問題ありませんが、LIST3:16:G で問題もなるのは
clrscr(); class that the class of the class
          dotcrsoff();
          sideclr(399); a a a a allowed little that out A a act of the table
termkey(); では、 LIST3-16.Cでは比較的自由自体を指する。
           textcursor (DISP CURSOR);
開数 getname を用棄ました。特徴はよっ変年ポラキー情報参数り込ん
void fend(void)
           dispwaku();
  initcolor();
          textcursor (NODISP CURSOR):
  initkey();
          outportb(0x64, 0);
/* 左側画面の消去 */
void sideclr(int y)
                               if (bloadskey(fkeyinf) & 2 SRIFT) break. <a. rib> ebulonit
          vramdotset(0, 0, 0xff, 1, 2, 4, 8, 36, y, 0, 0);
/* 編集エリアのサイス変更用 */
void change box (void)
          fstart();
          ex = 0; ey = 0;
           initset():
           fend();
```

■ ディレクトリ内のファイルの探索 これの エススス のようのから

ファイル名にワイルドカードが入力された場合には、指定したワイルドカードに従って画面上にファイルを表示させなければなりません。というわけで、この処理を実現しているのが file _disp()という関数です。この関数の流れは、次のようになっています。

- 1. 関数 findfirst で、初めに一致するファイル名(ワイルドカードを 含む)を探索し、存在すれば画面にそのファイル名をプリントす る。
- 2. 関数 findnext で、次に一致するファイル名を探索する。一致するファイルが存在すればプリントし再び次に一致するファイル名を探索する。もし、一致するファイルが存在しなければ処理を終了とする。

ここでのポイントは、ファイル名の入力と、ディレクトリ内のファイル表示、ファイルポインタによるファイルの入出力管理の3つにしぼられます。この3つさえ理解できれば LIST3-16.C は卒業です。では実際にこれらを組み込んだプロジェクト LIST3-18.PRJ を実行してみましょう。なお、グラフィックデータをロードする時の中断は [SHIFT] キーとしました。

	LI	ST3-18.C	or (blow) daste	blov	nzedzo
#include <graphics.< th=""><th>h></th><th></th><th>(biov) bne</th><th></th><th></th></graphics.<>	h>		(biov) bne		
#define MAKE_CR	0x39		hange bdk(vok		
#define MAKE_CL	0x38				
#define MAKE_F1	0x62				
#define MAKE_F2	0x63				
#define MAKE_F3	0x64				
#define MAKE_F4	0x65				
#define MAKE_F5	0x66				
#define MAKE_SHIFT	0x70				
#define MAKE ZOOM	0x29				

```
#define MAKE ROLL
              0x13
              0x3c 深刻のパトナでの内はイケイト等
#define MAKE RIGHT
#define MAKE LEFT
              0x3b
#define MAKE UP
              0x3a
#define MAKE DOWN
              ファイル名にワイルドカードが入力された。 bEx0
#define MAKE HANTEN 0x16
#define MAKE LOAD 0x25
#define MAKE SAVE
              0x1e
#define MAKE CHANGE 0x2b
extern void grphinit (void);
extern void pointer_set(void);
extern void zoombitdt (void);
extern void dispwaku(void);
extern void initset (void);
extern void dotcsr(int);
extern void initcsr(void);
extern void termcsr(void);
extern void initkey (void);
extern void termkey(void);
extern void rcolor(void);
extern void lcolor (void):
extern void initcolor(void);
extern void allpaint(int, int, char, int);
extern void ptn_copy(void);
extern void rolling (void):
extern void dotshiftr(void):
extern void dotshiftud(int);
extern void udhanten (void);
extern void gsave (void) ;
extern void gload(void);
extern void fstart (void):
extern void fend(void):
extern void change box (void);
extern int
  wwindow,
  color,
  kevdat.
  space;
```

```
LIST3-18 POLI:MOOS_BXAM GERO
int xmax = 64,
  ymax = 64,
  xmin = 0,
  vmin = 0;
void main()
(11st3-6
grphinit();
  pointer set();
  initset();
  dispwaku();
 zoombitdt();
 initcsr();
  initkey();
  initcolor();
keydat = 1;
  while (keydat != 0) {
      switch (keydat) {
case MAKE CR:
            rcolor();
            break;
  case MAKE CL:
            lcolor();
            break;
  case MAKE_F1:
            allpaint (wwindow, color, 0xff, 0);
            break;
         case MAKE F2:
            allpaint (wwindow, color, 0xaa, 1);
            break:
         case MAKE F3:
            allpaint (wwindow, color, 0x55, 1);
            break;
         case MAKE F4:
            allpaint (wwindow, color, 0xaa, 0);
            break;
         case MAKE F5:
            allpaint (wwindow, color, 0x55, 0);
            break:
         case MAKE SHIFT:
            ptn copy();
            break:
```

```
case MAKE ZOOM:
            zoombitdt();
              break;
                   case MAKE ROLL:
                   rolling();
                   break;
                   case MAKE RIGHT:
                   dotshiftr();
                  break;
                              case MAKE LEFT:
                  dotshiftl();
                   break;
                 case MAKE UP:
                 dotshiftud(0):
                 break:
                 case MAKE DOWN:
                dotshiftud(1):
                  break;
                 case MAKE HANTEN:
                 udhanten();
                break;
                case MAKE LOAD:
                 fstart();
                wold all gload();
                fend();
                wold well break; (Affin) ,zoloo ,wobniww) inisgils
                case MAKE SAVE:
                wold dot gsave(); sak0 , coloo ,wobchww) dalagiis
                fend();
                break;
                  case MAKE_CHANGE: TO TOO GOOD THE STATE OF T
                change box();
                break;
                   default : wasan rolos wabalww salsgifa
                                         break;
 }
       }
       termkey();
       termcsr();
       closegraph();
```

	LIST3-18.F	PR.I	THRI_IMAN	
14-+2-0	fetack (vold):			
1io+2-1				
list3-0				
11513-2				
TTOCO I				
list3-6				
list3-8				
list3-10				
list3-12				
list3-14				
list3-16				
list3-17				
list3-18				

本書ではサンプルプログラムということで機能的には、かなり省いたものとなってしまいました。

しかし、これだけの機能があれば、本書で作成するゲームでは十分といえるでしょう。最後に、パターンエディタの締めくくりとして2章で作成した、マウスによる、お絵描きツールとドッキングさせてみましょう。当然のことながら、マウスとマウスドライバ(mouse.sys)が必要となってきます。では、メイン関数とプロジェクトファイルを打ち込んで起動してみてください。

		LIST3-19.	С			nxedse
#include <graphics.< td=""><td>h></td><td></td><td></td><td>rcolor(void);</td><td>blov</td><td>nzedze</td></graphics.<>	h>			rcolor(void);	blov	nzedze
#include <dos.h></dos.h>						
						extern
#define MAKE_CR	0x39					
#define MAKE_CL	0x38					
#define MAKE_F1	0x62					
#define MAKE_F2	0x63					
#define MAKE_F3	0x64					
#define MAKE_F4	0x65					
#define MAKE_F5	0x66					
#define MAKE_SHIFT	0x70					
#define MAKE_ZOOM	0x29					
#define MAKE_ROLL	0x13			cursorkor(int,		
#define MAKE_RIGHT	0x3c					

```
#define MAKE LEFT 0x3b
#define MAKE UP
                0x3a
#define MAKE DOWN 0x3d
#define MAKE HANTEN 0x16
#define MAKE LOAD 0x25
#define MAKE SAVE 0xle
#define MAKE CHANGE 0x2b
extern int
   wwindow,
   mwindow.
   color,
   keydat,
  memory adr,
   msonoff.
 tobit_adr, and submitted to Aud 4 A E V of the Kentleman
   csr on,
   space;
extern void grphinit (void);
extern void pointer_set(void);
extern void zoombitdt(void);
extern void dispwaku (void) :
extern void initset (void);
extern void dotcsr(int);
extern void initcsr(void);
extern void termcsr(void);
extern void initkey (void);
extern void termkey (void);
extern void rcolor (void);
extern void lcolor(void):
extern void initcolor (void);
extern void allpaint(int, int, char, int);
extern void ptn copy(void);
extern void rolling (void);
extern void dotshiftr(void);
extern void dotshiftl(void);
extern void dotshiftud(int):
extern void udhanten (void) :
extern void dot plot(int);
extern int xycset(int, int, int, int);
extern void cursorxor(int, int, int);
```

```
extern void gsave (void);
extern void gload (void);
extern void fstart (void);
extern void fend(void);
extern void change box(void); by yromem) rox108180
extern void file disp(void);
extern void mouseinit (unsigned int, unsigned int,
             unsigned int, unsigned int);
extern void mouseterm(void);
extern void mousexyset(int, int);
int xmax = 64, a MARK BEATER
   ymax = 64, same(testing actor wobstew) integffs
   xmin = 0,
   ymin = 0; ass MARR MARR
void main()
1
   int sfsin; askakkaddan , rolog (wobniwe) jnisqila
   grphinit();
   pointer_set(); %(0) pased , roloo , wobalww) salaqiis
    initset();
   mouseinit(mwindow * 8, 639, 0, 399);
    cursorxor(memory_adr, bit_adr, color);
    dispwaku();
    zoombitdt();
    initcsr(); if (one on ** 1)
    initkey();
    initcolor(); a calme sld , the vacmem) toxtosauc
    keydat = 1;
    while (keydat != 0) {
       switch (keydat) {
           case MAKE CR:
              disable();
             if(csr on == 1)
          cursorxor(memory_adr, bit_adr, color);
              msonoff = 0;
              enable();
           rcolor();
              cursorxor(memory_adr, bit_adr, color);
              msonoff = 1;
```

```
break;
case MAKE_CL:
                                                                                 (biov) baole blov mrsiks
 disable();
                                                                                extern void fetart (void);
if(csr on == 1)
                                                                                  extern void fend(void);
 cursorxor(memory_adr, bit_adr, color); low masked
 msonoff = 0;
    marine HARA (H enable(); opieno (ini benpienu ) ilniesuom biov nieske
                             lcolor(); and and the beaplease
                           cursorxor (memory adr, bit adr, color); and blow and the
        msonoff = 1; (sal (sal) seeyxeesom blow are need
                             break:
        case MAKE F1:
        allpaint (wwindow, color, 0xff, 0);
       break;
       case MAKE F2:
       allpaint (wwindow, color, 0xaa, 1);
                           break;
       case MAKE F3:
        allpaint (wwindow, color, 0x55, 1);
      break:
      case MAKE F4:
        allpaint (wwindow, color, 0xaa, 0); () jee setalog
             break;
        case MAKE F5: 1000 10 000 18 Wobnium) Siniagon
        allpaint(wwindow, color, 0x55, 0);
  break;
case MAKE SHIFT:
      disable();
      if(csr_on == 1)
   desa wold realer (cursorxor (memory adr, bit_adr, color); assist
      msonoff = 0;
enable();
extern void all ptn copy(); to char, int) | (Jabveshinbaiwe
msonoff = 1;
break;
case MAKE ZOOM: (1 == no men) 11
zoombitdt(); bs yromen) zoxiosius
break;
case MAKE ROLL:
rolling();
break; when the description of t
case MAKE_RIGHT:
```

```
dotshiftr();
          break;
       case MAKE LEFT:
          dotshiftl();
          break:
       case MAKE UP:
dotshiftud(0);
          break;
       case MAKE DOWN:
          dotshiftud(1);
          break:
       case MAKE HANTEN:
          udhanten();
          break;
       case MAKE LOAD:
    disable();
          if(csr on == 1)
              cursorxor(memory_adr, bit_adr, color);
          msonoff = 0;
          enable();
          fstart();
          gload();
          fend();
          msonoff = 1;
          break; A A A A A A A
       case MAKE SAVE:
          disable();
          if(csr on == 1)
              cursorxor (memory adr, bit adr, color);
          msonoff = 0;
          enable();
          fstart();
          gsave();
          fend();
          msonoff = 1;
          break;
       case MAKE CHANGE:
          change box();
          break;
       default :
          break:
```

```
mouseterm(); MARE CD: (() Illinated (); termkey(); termcsr(); (osegraph(); (osegrap
```

```
LIST3-19.PRJ
1ist 2-0
list2-1
list2-6
list2-9
list2-11
list3-0
list3-2
list3-4
list3-6
list3-8
list3-10
list3-12
list3-14
list3-16
list3-17
list3-19
```

```
case MAKE_SAVE:

disable();

if (car_on == 1)

cursorxor(memory_a8i, "bis_save |

cursorsor(memory_a8i, "bis_save |

cursorsor(memory_a8i, "bis_save |

cursorsor();

cursorsor();

cursor();

cursor();
```

break) .ccxD

第4章 TVゲームの世界

グラフィックス処理のひとつの分野として発展してきた TV ゲームですが、一口にゲームといっても、迷路型、シューティングタイプ、ロールプレイングなど様々な形態があります。中でもリアルタイムゲームの分野は、プログラムの結果を確かめながら進めることができるために、プログラミングも楽しく進めていくことができます。本章では3章のパターンエディタでデザインしたキャラクタを使って、実際にアニメーション処理を、さらに、これを発展させたリアルタイムゲームを作っていきます。

まず話を進める前に、アニメーション用の口絵パターンを用意しておきましょう。デザインしたパターンは、ひとつのファイルとしてカレントディレクトリへ格納しておいてください。プログラムを効果的に、また楽しく進めていくためにも、このような地味な作業が必要となってきます。もちろん、パターン数さえ揃っていればオリジナルのデザインでもなんら差し支えありません。なお、各キャラクタの大きさは32×32ドットとしました。

アニメーション処理の実際は、説明するまでもなく徐々に動いていく過程を連続して表示すれば実現できるわけです。対象となる動作をいかに細かく分解するかによって、動きのなめらかさが決まるのです。もっとも、メモリに制限のあるパソコンでは、見せ場となるパターンにより多くの動作パターンを割り付けるという手法をとります。

さて、デザインしたパターンを実際に利用するには、一枚の絵となっている画面上の絵を適当な大きさに切り取ってメモリへと格納しておかなければなりません。TURBO C には画面上の指定された範囲のビットイメージをメモリへセーブする getimage()という関数と、セーブしたビットイメージを画面上に復元する putimage()という関数があります。これらの関数を利用するのもひとつの方法としてあげられます。

しかし、TV ゲームなどでは画面上のビットイメージを単純にセーブしてそのまま復元するよりも、これらのビットイメージをさらに加工する場合がしばしばですから、構築するシステムに合わせた独自の関数を作る必要があります。そこで、これまでの応用も兼ねて、画面上のビットイメージをメモリへセーブする関数と、画面に絵を復元する関数を作ることにしました。

メモリとは、すなわち変数のことですが、ここで画面上のデータを変数

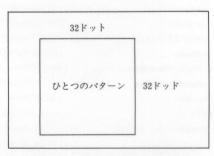


図4-1 パターンのデータ構造

へと格納することを考えてみます。口絵のデザインではひとつのパターンの大きさを32×32ドットしましたから、この場合のデータ構造は図4-1のようになります。

まず X 方向です、これは汎用性を考慮すれば char 型の変数配列を使いたい所ですが、スピード最優先のリアルタイムゲームということで、データをまとめて処理できる32ビット長の long 型の変数を使うことにします。ですから、X 方向は long 型の変数ひとつで対応できるわけです。次に Y 方向ですが、これは単純に long 型の変数が32ライン重なることになります。また、プレーンは4つありますから、ひとつのパターンを構成する変数域は次のように、簡単な構造体として定義できることになります。

また、実際にはパターン数がひとつではありませんから、この構造体自体を配列としてパターン数分確保することにします。

ここでは、50個のパターンを格納できる構造体を定義しましたが、この 数は適当に設定してください。

さて、このように変数の構造が確定できれば、あとは単純に画面上のデー

タを取り込むだけです。早速、ビットイメージをメモリヘセーブする pattern_in()という関数と、処理としては逆のビットイメージを画面上に復元する pattern_out()という関数を作ってみましょう。

```
LIST4-0.C
#include <stdlib.h>
#include <stdio.h>
#include <comio.h>
#include <dir.h>
extern char far *c blue;
extern char far *c red;
extern char far *c green;
extern char far *c itsty;
#define PTY 32
/* パターン格納用構造体の定義 */
struct ( A & M ( M ) - THE WANTE AVER A [FT] dg prof
   long dtb[PTY];
  long dtr[PTY];
 long dtg[PTY];
   long dti[PTY];
} pattern[50];
/* 画面上データをメモリへセーブする */
pattern in (unsigned int i, int j)
{
   int k;
   if(i < 50) {
      for (k = 0; k < PTY; j+=80, ++k) {
          pattern[i].dtb[k] = *(long far *)(c blue + j);
          pattern[i].dtr[k] = *(long far *)(c red + j);
          pattern[i].dtg[k] = *(long far *)(c_green + j);
         pattern[i].dti[k] = *(long far *)(c_itsty + j);
       i = 0;
   else i = 1; 宗公科芸術 3 名 7 内部 3 < 一 4 / 0 图 0 3 、 1 5 5 5
   return (i);
 のように多数時間管が維持で変われて、野球はよ真顔に補間との予
```

```
/* メモリから画面へパターンを復元する */
pattern_out(unsigned int i, int j)
{
    int k;
    if(i < 50) {
        for(k = 0; k < PTY; j+=80, ++k) {
            *(long far *) (c_blue + j) = pattern[i].dtb[k];
            *(long far *) (c_red + j) = pattern[i].dtr[k];
            *(long far *) (c_green + j) = pattern[i].dtg[k];
            *(long far *) (c_itsty + j) = pattern[i].dti[k];
            *
            i = 0;
            }
            else    i = 1;
            return (i);
}
```

これら関数の引数には、パターン NO.と 32×32 ドットで構成する BOX 型の左上の G.V.RAM アドレスを与えることとしました。なお、パターン 数の最大値である50を超えた場合には、なにも処理せずに関数の戻り値として1を返すようにしてあります。また、パターン NO.を示す引数iの定義では unsigned としていることに注意してください。このことにより配列を参照する時のiの範囲は $0 \le i < 50$ と確定されるのです。if 文の条件判断を少なくするささやかな配慮といえます。

#include <graphics.h>
#include <conto.h>
#include <stdio.h>
#define NPAT 80*32 LST.1-1781J

#define NPAT 80*32 extern int pattern in (unsigned int, int);

extern int pattern out (unsigned int, int);

extern int gload (void);

■ キャスト演算子

LIST4-0.Cでは、配列を参照するための変数の範囲に気を配ると共に、 もうひとつ重要なキャスト(cast)という概念が導入されています。

キャストとは、いったん定義した変数や式の型を、一時的に別の型へと変換することを指しています。LIST4-0.Cでは、char型のG.V.RAMへのポインタの型をパターンを格納してある変数の型である long 型へとキャスト(型変換)しています。このキャストの特徴は、型を変換するまでは本来与えられた型が有効であるということです。当り前のようですが、このことが重要なのです。

LIST4-1.C では、これを利用してフィールドのアドレス計算をする時には本来の char 型で、データのやり取りは long 型というように、最も細かく G.V.RAM を操作できる char 型の特徴を生かしながら処理の高速化を図ったというわけです。

では実際に、これらの関数を利用してアニメーション処理を実行してみましょう。main()関数とプロジェクトファイルは次のようになります。

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#define NPAT 80*32

extern int pattern_in(unsigned int, int);
extern int gload(void);
int mwindow = 36;
```

```
DISTANCE CARLS & FIRES J. LEAT CH
void main()
           ピケメッセージが現れます。ファイル名を指定する
  int i, j, k, l, m, n;
  grphinit(); ( TYE at & Jtt & ON 4 X I S. ON 4 - X X ON
  pointer_set(); Ask of district the kind and the Lord & Jo
  clrscr();
 if(gload() == 0) {
    for (k = m = n = 0, l = mwindow; k < 4; ++k, l+=NPAT) {
  for (i = 0, j = 1; i < 11; ++i, j+=4) {
         if(n != 0) break;
は基本と多りでも延伸な鍵盤に、化かびも進出が挙行を発伸が不順軟化デルス
if(n != 0) break;
 マップ clrscr():アーンは他のパターンと同様に32×62がよびはままのと
    printf("\n\15-\ \n' 5-\/ no.=");
printf("I); / / /- / no.=");
    scanf("%d", &j);
           単するものとします。ですからマップデータを管
スカの i = k; 次のように単級か2次元級別とすればよいでもょう。
    do f
 pattern_out(i++, 0);
 if (i > j) i = k;
 } while(getch() != 0x1b);
 else (ップエディタを作る前に、ここで定義したデータをディスク
    printf("Yn press any key");
          する為の専用の関数を作ることにします。
   getch();
  closegraph();
```

```
LIST4-1.PRJ
```

```
list2-0
list2-1
list3-16
list4-0
list4-1
```

LIST4-1.Cを実行すると、パターンが格納してあるファイル名の入力を 促すメッセージが現れます。ファイル名を指定すると、次にアニメーショ ンのスタート NO.とエンド NO.を入力します。 コマ送りは[SPACE] キー としました。では、口絵のパターンで実験してみましょう。アニメーショ ンのスタート NO.として7をエンド NO.として18を入力してください。 [SPACE] キーを押し続けると三角形の板が回転している状態が表示され るはずです。この例ではパターンの大きさを32×32ドットとしましたから 迫力に欠けますが、もっと大きなパターン用の関数を作って試してみるの もおもしろいでしょう。また、セーブしたパターンを斜め方向に復元した り、構成する色を変えたり等、セーブしたビット情報を色々と加工してみ るのもよいでしょう。

■ マップエディタに挑戦

さて、ここで LIST4-0.C のおもしろい応用を考えてみましょう。それは、CRT 画面の中に2次元の世界を作り出すためのマップエディタというツールです。マップエディタというのはパターンエディタと並んで TV ゲームには欠かせないツールとなります。しかも、LIST4-0.C を使えば簡単に作ることができるのです。

さて、複雑なマップを作るにはそれなりに複雑な処理となりますが、こ こでは移動方向は一方向のみという制限を付けることにします。

マップの基本パターンは他のパターンと同様に32×32ドットとすると、 画面横方向には20個のパターンを並べることができますから、マップデー タはパターンが横に20個並んでおり、それが縦方向に展開する構造とし、 各パターンは番号で管理するものとします。ですからマップデータを管理 する為の変数は、次のように単純な2次元配列とすればよいでしょう。

```
#define PYMAX 240-1
#define PXMAX 20-1
unsigned char map_data[PYMAX+1][PXMAX+1];
```

まず、マップエディタを作る前に、ここで定義したデータをディスクに セーブしたりロードする為の専用の関数を作ることにします。

```
/* マップデータ格納用変数配列の定義 */
unsigned char map_data[PYMAX+1][PXMAX+1];
/* マップデータをディスクヘセーブする */
map_data_save(void) {
        char fname [64];
  FILE *sfp:
      char c;
        int i, j, k, l, fsin;
        vramdotset(0, 0, 0xff, 1, 2, 4, 8, 36, 400, 0, 0);
        fsin = 0;
   while(1) {
         while(i)
   printf("\forall n\forall 
                     i = getname(fname);
if(i == 1) {
   fsin = 2;
                                break;
                            いスターンは番号で管理するものとします。ですからマッ
       if(i == 2) file_disp(fname);
                        else i = 0;
                if(fsin == 2) break;
                if(access(fname, 0) == 0) { xxxxyq1xxxb qxx xxxb benpianu
                        printf( "Yn%s is allready exist Yncontinue (Y-N) ? ",
まず、マップエディタを作る前に、ここ:( mane) データをディスクル
                        while(1) {
                     で = getch(): さんの専用の関数を (()
                                if(c == 'n' || c == 'N') {
                                        fsin = 2:
                                      break;
                                printf("Y");
                              /* 11st3-16.c 0420 N-F #/
                        if(fsin == 2) break;
                else printf("save file name = %s ", fname);
```

```
* Analysis sfp = fopen(fname, "wb"); allgalb ell? (S == 4/1)
if (NULL == sfp) {
Sinclude (difsin = 3;
#inglude <iobreak; | placed (S == nle3)31
fsin = 0;
for(i = 0; i <= PYMAX; ++i) {
if(fputc(map data[i][j], sfp) == EOF) {
#define DOWN '2' fsin = 1;
Adefine ESC Oxid break;
#define SPACE '}' | (1++ ;XAMY9 => 1 ;0 = 1) rol |
#define LOAD } '1' | ( [++ (XANX9 => [ \0 = [] xo)
#define LOAD if (fsin == 1) break; ((qts) prept = 0)) hi
#define NOAD3 '3'
fclose(sfp);
break;
map data(1)(1) = c; ESVAS on look
fmes(fsin, fname);
return(fsin);
#define XMAX2 20-1
/* マップデータをディスクからロードする */
map data load(void)
 (define DOMING 0
char fname[64];
FILE *sfp;
int c, i, j, k, l, s, done, fsin;
           struct ffblk ffblk;
            vramdotset(0, 0, 0xff, 1, 2, 4, 8, 36, 400, 0, 0);
       | fsin = 0; 大き内容を発表を表がするパターン番号を格的とた;
           while(1) {
            :格納してあるパターン番号から、画面を復元する機能を表すれる。
            off of the state o
printf("\forall n\forall n\for
                  printf("Yninput map data file nameYn");
i = getname(fname);
if(i == 1) {
extern int map fsin = 2; (void);
                                           break:
wold right sdt (wold) /
```

```
unstoned the else i = 0; wax attraway att 1 (qls == LIUM) li
  if(fsin == 2) break;
 printf("load file name = %s ", fname);
  sfp = fopen(fname, "rb");
  char c; fsin = 3; } (t++ :xanxq => t :0 = t) xo2
      break; E ... (gla .([][l]sisb qsm) big3) 11
  }
  wear fsin = 0; o ours 1 2 4 8 34 140014 as
  for(i = 0; i <= PYMAX; ++i) {
  for(j = 0; j \le PXMAX; ++j) {
     if((c = fgetc(sfp)) == EOF) {
             fsin = 1;
     select " break; as save file manager, t (qle) esolol
        i = oluname (finame);
        map data[i][j] = c;
       } fain = 2/
        if(fsin == 1) break;
      fsin = 0;
      fclose(sfp);
     break:
   fmes(fsin, fname); 83 000 0) (100) oman1 dano
   return (fsin); ( "Yets is allready exist Yedonkinus 1914) $117
```

マップエディタというのは、基本的にマップデータ用に定義した配列へ、 画面に展開したパターンに対するパターン番号を格納したり、逆に、配列 に格納してあるパターン番号から、画面を復元する機能を持たせればよい わけです。では、プロジェクト LIST4-3.PRJ まで進んでください。

```
LIST4-3.C (1 = 1) 11

/*
マップエディタ
*/
#include <graphics.h>
#include <conio.h>
```

```
#include <stdio.h>
#include <bios98.h> breaks (blov) is nwob blov
#include <dir.h>
#include <io.h> left set(); (ini) feerwoxed blov
#define NPAT
            80*32
#define RIGHT '6' up set(); .0 * SE * 08 - 0000x0 = wobniwqm
#define LEFT
           '4' break: ,I = ini xod
#define UP
            181
#define DOWN
            '2' down sat ()
#define ESC
            0x1b
#define SPACE
           Tolow SPACE:
#define LOAD1
#define LOAD2
            'L'
#define LOAD3
            11] 1
#define SAVE1
           's' map data (ny) (px) * (char) (x + y * (youtsbamph);
#define SAVE2
            151
                           /* マップエディを用メイン関数の定義 */
#define SAVE3
           1 1 1
#define XMAX 11-1
#define YMAX
            4-150 10002
            20-1s toanses type in ,m ,i ,d ,i ini
#define XMAX2
#define YMAX2
            6-1 map data losd(); ..... :() ilainqup
#define XMIN
            O boxcursel(y2 * 32 a White this thinks) Ligaise
#define YMIN
            0 disp mac(0); } (0 === ()bsolp)ti
#define XMIN2
#define YMIN2 0 * MANGARA A Supporting a do .0 aprox = m = M) wol.
#define PXMAX 20-1 (Armid as little Adda > de 14 armid a 0 pr. 14 gol
#define PYMAX 240-1 reals 1(t ++m) nl gredged = n
extern char far *c blue;
extern char far *c red;
extern char far *c_green;
extern char far *c_itsty; 0;
extern unsigned char map data[PYMAX+1][PXMAX+1];
extern int pattern_in(unsigned int, int);
extern int pattern_out(unsigned int, int);
extern int gload(void); 9 + 4 * Sx + 08 * SE * Sy) (carupxod
extern int map_data_save(void); (Dax =1 yex) slidw
extern int map_data_load(void); ()dojep = yel
void right set (void);
```

```
void down_set(void);
void up set (void);
void boxcursol(int);
void disp map(int); file name w to ", forme);
int mwindow = 36,
        mowindow = 0x7d00 - 80 * 32 * 6,
        box inf = 1,
        \mathbf{x} = \mathbf{0}
       y = 0,
        x2 = 0,
        y2 = 5,
        px = 0, 000 (4 m a) 4 cm pysky; 4449
        py = 0, if({e = fgetc(afp)} == EGF) ( -- 'I' SGAOI enliebt
        py0 = 0,
       boxadr:
/* マップエディタ用メイン関数の定義 */
void main()
        int i, j, k, l, m, n, key;
        grphinit();
        pointer set();
        printf("\Yn" \quad \f' - \quad \Pi - \quad
        if(gload() == 0) {
        for (k = m = n = 0, l = mwindow; k <= YMAX; ++k, l += NPAT) {
        for (i = 0, j = 1; i < 11; ++i, j+=4) { or xaxx and and above
                                 n = pattern_in(m++, j); 1-025 xaxxq enliabt
                                  if(n != 0) break;
                      if(n != 0) break;
    に展開したパターンに対するパターン器局が破倒られるt xailolitaotice
                 clrscr();
                key = 0;
                pattern_out(x + y * (XMAX + 1), 0);
                 disp_map(0); (int ,int bengitem) ni_mretisq int mretks
                 boxcursol(y * 32 * 80 + x * 4 + mwindow);
                 boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
                 while (key != ESC) {
      way x # key = getch(); (blow) bad asab gam snl massxe
                    switch(kev) {
case RIGHT:
```

```
right_set();
              ifty o break; " warma ("yest you seeing nk") linking ....
                              case LEFT:
             patrary our break; - (wax + 1) 0;
        case UP:
       up_set();
              baseurselly break; * 55 + 32 * \* 概题の密想は音机く一枚又寸を除 */
               case DOWN:
              down_set();
                                    break; \tanker \tanker
              boxcurs case SPACE: 30 4 x2 4 4 sewindas # bomo Ralyas
                                     boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
                           pattern_out(x + y * (XMAX + 1),
                                                              y2 * 32 * 80 + x2 * 4 + mpwindow);
     map data[py][px] = (char)(x + y * (XMAX + 1));
boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
                                      break;
       KEY IMPO ke case LOAD1: XAMY = y (XAMY < y) 11
       case LOAD2:
       LE (blood Note case LOAD3: W # A # 08 * SE * y) Loss word
               map data load();
                                      py0 = 0;
             boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
              boxes and (w disp map (0); Sx & Gast A&A Sy) Losausxod
              boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
              pattern out (x + y * (XMAX + 1), 0);
              case SAVE2:
               case SAVE3:
                                   map_data_save();
                       --py0: py0 = 0; /* 施馨の小詞花式ポソーセスサモザ */
                     boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
                      disp map(0);
                                   boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
               pattern out (x + y * (XMAX + 1), 0); mo langed
               break;
               default : break; %2x4400 modified ) formoxod
```

```
else (
printf("Yn press any key");
getch();
void becomment (int); -
closegraph();
lot mwindow - 36.
/* ボックスカーソル右方向への移動 */
void right set (void)
( w = 0.
   KEY INFO keyinf;
  keyinf.cmmd = 2;
   if (bios98key(&keyinf) != 0) {
   boxcursol(y * 32 * 80 + x * 4 + mwindow);
  v2 * 32 * 80 * x += 1; 3 * 5x + 08 * 52 * 5v
  if (x > XMAX) { (gade) = [xq] [vq] sdah qam
   (wobal x = XMIN; Sx + 08 * SE * Sy) losanoxod
  y += 1;
  if (y > YMAX) y = YMAX; IGAO1 8885
   boxcursol(y * 32 * 80 + x * 4 + mwindow);
   pattern_out(x + y * (XMAX + 1), 0);
   else {
   boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
   ++x2; # # F # Cx # 00 " EE M Cy) for wood 1 4 4 x 1 4 4
   if(x2 > XMAX2) x2 = XMIN2;
      px = x2; a = pattern in(s++, j); cseerd
    boxcursol(v2 * 32 * 80 + x2 * 4 + mpwindow);
/* ボックスカーソル左方向への移動 */
void left set (void) * Cx * 08 * SE * Syllogaroxod
   KEY INFO keyinf; Sa + OB - St + Syllowship
keyinf.cmmd = 2;
   if(bios98key(&keyinf) != 0) {
      boxcursol(y * 32 * 80 + x * 4 + mwindow);
      x -= 1;  getch()/
      if(x < XMIN) {
         x = XMAX;
```

```
/* ポックスカ y -= 1 (大 ボックスカーフル上方向への移動 *)
  if (y < YMIN) y = YMIN;
   boxcursol(y * 32 * 80 + x * 4 + mwindow); Talvest OTHI YER
  lf(box inf -- 1) (
  else ( - (year - Owobrithen + * x + 08 * SE * y) Loszuoxod -
      boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
      --x2; sevrands (k + 1, 0x66, 1) = xXAMX = y (MIMY > y) 11
      px = x2;
      boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
}
/* ボックスカーソル下方向への移動 */ (SMIMY > Sv) 31
void down set (void)
  KEY_INFO keyinf; 12 - XAMY9 = Dyq (2 - XAMY9 < Dyg) %1
keyinf.cmmd = 2; (0yg)qsm gaib sale
  if(bios98key(&keyinf) != 0) {
      boxcursol(y * 32 * 80 + x * 4 + mwindow);
   ++y;
   if (y > YMAX) y = YMIN; a ca a ca a ca a cy loerwoxed
     boxcursol(y * 32 * 80 + x * 4 + mwindow);
      pattern_out(x + y * (XMAX + 1), 0);
   else ( leste 4 is to 1)
      boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow); al) gam galb blow
      ++y2;
      if(y2 > YMAX2) {
         Tor(3 = 0, k = 0x7d00 - 32 * 80; 1 < 6; ++1) | ; Sy-
         --py0; ) (1-+1 , 1+6 +XANK 8 07 t 1x = 1 ,0 = 1)301
      if(py0 < 0) py0 = 0;
         else disp map(py0);
list2-1
  --ру;
if (py < 0) py = 0;
boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
```

```
/* ボックスカーソル上方向への移動 */
void up set (void) press any key") anthy = v (MIMY > v) 11
   KEY INFO keyinf; (wobalwa + 4 * x + 08 * SE * y) loszuoxoo
   keyinf.cmmd = 2; 4(0 (1 + XAMX) * ( + x) 300 assstag
   if(bios98kev(&kevinf) != 0) {
       boxcursol(y * 32 * 80 + x * 4 + mwindow);
  boxcursoily2 * 32 * 80 + x2 * W 4 6 6 6 6 6 6 7 7 - 7 4 6
    if(y < YMIN) y = YMAX;
       boxcursol(y * 32 * 80 + x * 4 + mwindow); 300 31
   pattern out(x + y * (XMAX + 1), 0);
   boxcursol (y2 * 32 * 80 + x2 * 4 + apwindows of lower laive
   else ( Odkay (akay int) 1 m 0) (
       boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
      if(y2 < YMIN2) {
          ++v2;
          ++pv0;
          if (py0 > PYMAX - 5) py0 = PYMAX - 5; tankyes OMMI YAM
          else disp map(py0);
       if (blos98key (skeydminites 0) by * s + 00 * 55 * y) longuesor
       ++py; "0 out by wolan Mark # 18 M on * 32 * g) losaus cod
       if (py > PYMAX) py = PYMAX;
       boxcursol(y2 * 32 * 80 + x2 * 4 + mpwindow);
/* マップの表示 */
void disp map(int py0)
   int i, j, k, 1;
   for (i = 0, k = 0x7d00 - 32 * 80; i < 6; ++i) {
for (j = 0, 1 = k; j \le PXMAX; ++j, 1+=4) {
pattern out((unsigned int)map data[i+py0][j], 1);
   k = 32 * 80;
   heyinf.comd = 2:
void xorvramdt(int, int, int);
```

```
/* ボックスカーソルの表示 */
void boxcursol(int k) - A A A A A A O V - T T V T S #00 PO 3 A
int xmax = 32, ymax = 32;
  int i, j; 在大脑中的 数据图图 DE 文章士是任意对多多多位的对象
  if (box inf == 1) {
 j = (ymax - 1) * 80;
  for(i = 0; i < xmax / 8; ++i, ++j) {
        xorvramdt(k + i, 0xff, 1);
     xorvramdt(k + j, 0xff, 1);
     j = xmax / 8 + k + 79;
k += 80;
 xorvramdt(k, 0x80, ymax - 2);
     xorvramdt(j, 0x01, ymax - 2);
1し丁表示する機能があって、これをうまく利用する。
  else box_inf = 1;
/* 指定アドレスへ指定データを書き込む */
void xorvramdt (int i, int j, int k)
  int 1;
  for (1 = 0; 1 < k; ++1, i+=80) {
     *(c blue + i) ^= j;
     *(c red + i) ^= j;
     *(c_green + i) ^= j;
    *(c_itsty + i) ^= j;
```

```
LIST4-3.PRJ

list2-0
list2-1
list3-0
list4-0
list4-2
list4-2
```

このマップエディタで注意しなければならないのは、マップデータの終わりの判断をどのようにするかです。マップデータの変数は unsigned

char としましたから、格納できる数値は0~255となります。そこで、本書では、この中の0番をマップデータの終わりを表す番号とすることにしました。ですから、パターン番号の0番はマップを構成するパターンとしては使えないということになります。なお、この関数の先頭では、マップデータを0で初期化をしています。

このマップエディタで注意しなければならないのは、マップデータの終

■ GDC によるスムーズスクロール

マップが完成したなら、次に画面のスクロールにチャレンジしてみようではありませんか。この画面スクロールは TV ゲームに限らずグラフィックス処理には欠かせないテクニックであるといえます。さて、このスクロール処理には色々な方法がありますが、ここでは標準的な GDC (Graphic Display Controler) による方法を紹介しましょう。

GDCとは、PC-9801シリーズの初代から登載されているグラフィック画面コントロール専用のLSIです。このLSIが持っている機能には画面を分割して表示する機能があって、これをうまく利用することで画面のスクロールが可能となるのです。このスクロールはハード的なスクロールであることから、ハードウェアスクロールともいわれます。図4-2を参照してください。

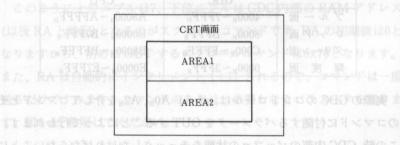


図4-2 GDC による分割画面と CRT 画面との対応

GDCではこのように任意のアドレス(ワード単位)でひとつの画面を二分割して表示することが可能なのです。この時、表示開始位置は AREA1の 左上となります。また、この AREA1と AREA2の位置はワード単位で任意 に設定できますから、図4-3のように AREA1と AREA2の構成を連続的に変化させることによって、表示開始位置を徐々に移動することが可能となり、見かけ上スムーズなスクロールが実現できるのです。

さて、GDCのコントロールで分かりにくいのはアドレス管理の方法で

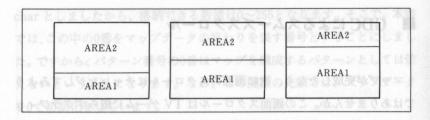


図4-3 GDCによる画面分割によるスクロール

しょう。GDC と CPU は同じ G.V.RAM をアクセスするのですが、割り付けてあるアドレスはまったく異なります。しかも同じ G.V.RAM であるにもかかわらずアクセス単位は CPU が8ビットであるのに対して、GDCは16ビットを単位としています。表4-1にそれぞれが管理しているアドレスを示しますから参考にしてください。

表4-1 GDC と CPU との G.V. RAM の対応

	GDC	CPU
ブルー面	$4000_{\rm H}\!\sim\!7{\rm FFF}_{\rm H}$	A8000 _H ~AFFFF _H
レッド面	$8000_{\rm H} \sim {\rm BFFF_H}$	$B0000_{H} \sim B7FFF_{H}$
グリーン面	$C000_{H} \sim FFFF_{H}$	B8000 _H ~BFFFF _H
輝 度 面	$0000_{\rm H}\!\sim\!3{\rm FFF}_{\rm H}$	E0000 _H ~E7FFF _H

実際の GDC のコントロールは、ポート $A0_H$ 、 $A2_H$ を介してコマンドとそのコマンドに付随するパラメータを OUT することによって行われます。この時、GDC 内部のバッファの状態をチェックしなければならないことに注意してください。 GDC では処理を効率よく進めるために16個の FIFO (First In First Out) バッファがあって、コマンドやパラメータを一時的に蓄えて実行する構造となっているのですが、バッファがいっぱいであればウェイトを取らなければならないからです。 GDC の状態はポート A0H から表4-2のようなステータスフラグとして得られます。

バッファの空き状態はビットの2でチェックすることができます (b2=1)。また、バッファの状態はビットの1でもチェックできますが、この場合にはコマンドやパラメータを送る前に一回一回チェックしなければなりま

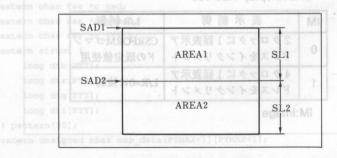
表4-2 GDC のステータスフラグ

ビ	B ッ字四層のREAZ 字域展示
"	内 容
1	
(0)	DATA READY
1	FIFO BUFFER FULL
2	FIFO BUFFER EMPTY
3	DRAWING
0 4	DMA EXECUTE
5	VERTICAL SYNC
6	HORIZONTAL BLANK
7	LIGHT PEN DETECT

せん。さて、コマンドですが、スクロールコマンドでは次のような構造と なります。

0	1	1	1	CAS RA →
Suffer !	10	0	0	SL2L

このように上位ニブルは7、下位ニブルは GDC 内部の RAM アドレス (以後 RA と略記)となるのがスクロールコマンドです。RA の初期値は0となりますから、初めに OUT するスクロールコマンドは0x70となります。また、RA は自動的にインクリメント (+1) されるので、コマンドは一度送るだけですみます。コマンドのあとは、必要となるパラメータを送ります。図4-4を参照してください。



EDIMENTO E

、下主 (1 公) 第一日年之

主电线设置

のコマンド

この間、GD

注意してく

(First in I

に蓄えて実

ばウェイド

このようにひとつの画面を AREA1と AREA2に二分割して、それぞれ の開始アドレスとライン数を SAD1、SL1と、SAD2、SL2とすると、各パ ラメータは表4-3のような順番で送ることになります。

表4-3 スクロールコマンドの各パラメータ

RA	7	6	5	4	3	2	1	0
0	+	版牌	NO	SAI	D1L	FRT	7 7 7	-
1	+	griss) Ti	rusi MTS	SAI	D1H	HOL:		4
2	-	SL	.1L	→	0	0	0	0
3	*	IM	←	Ne	SL	.1H	A R	→
4	+		C 5 /	SA	D2L			→
5	+	ye	- AND	SAI	D2H	T.V	G)	-
6	-	SL	2L	oriv	0	0	0	0
7	*	IM	+		SI	2H		\rightarrow

せいれた おっぽ

医电阻 建工厂

らないことに

*:DAD+2

DAD+2		0000ac	機	能	E0000 EZERE
0	``1"	によるイ	ンクリメ	ント	(DAD+1→DAD)
1	``2"	によるイ	ンクリメ	ント	(DAD+2→DAD)

DAD:Display Address

IM	表示制御	L/R 制 御
0	2クロックに1回表示ア ドレスをインクリメント	CSRFORMコマン ドの既定値使用
	4クロックに1回表示ア ドレスをインクリメント	L/R=0に強制

IM:Image

ところで、最近の機種ではGDCの動作クロックが5MHzと2.5MHzの 二通りがあり、ユーザーが選択して使用できるようになっていますから、

動作クロックについても考慮しなければなりません。どちらのクロックを使用するかはディップスイッチ2番の8によって選択します(OFFで2.5 MHz、ONで5MHz)。なお、変更に際してはシステムを再起動しなければなりません。

当然のことながら GDC のパラメータもこの動作クロックに合わせる必要があります。スクロールコマンドでは、2.5MHz で IM ビット=0、5MHz で IM ビット=1となります。では以上のことを踏まえて次のプログラムへと進んでください。

```
LIST4-4.C
/*
  GDCによるスムーズスクロール
#define PTY
          32
#define SCLDOT 2
#define SCLD16 SCLDOT*16
#define SCLPTC SCLDOT*40
#define GSTADR 0x4000
#define GENADR 0x7e80
#define STLINE 0x1900
#define ENLINE 0x1900-SCLD16
#define VEND 400*80
#define PYMAX 240-1
#define PXMAX 20-1
extern char far *c_blue;
extern char far *c red; *** | OTSIDS * AGATED * 158158
extern char far *c green;
extern char far *c itstv;
extern struct ( add.datab Dights and o) (* 181 pgol) *
  long dtb[PTY];
  long dtr[PTY];
  long dtg[PTY];
   long dti[PTY];
} pattern[50];
extern unsigned char map_data[PYMAX+1][PYMAX+1];
int enemy_table1[20]; _1++ _TOGIO8 > 1 +q = p .0 = 1) tol
int enemy sign;
```

```
char gdchz = 0;
void scroll (int gys, int gye, int sign)
   struct BYTE {
      char low;
char high:
HMa'n=4 w'Y MI
   union DATA (
short dataw;
      struct BYTE datab:
   1:
   union DATA
      nol,
      no2.
      ad1,
      ad2;
   static int
      sclad1,
      sclnol,
      sclno2,
      map ptn y,
      map y,
      map x;
   int 1, m, n, p, q;
   if(sign == 0) { /* sign=0ならば全画面をクリアする */
      map ptn y = PTY-SCLDOT;
      map y = 0;
      sclad1 = GSTADR + SCLPTC;
      for (1 = gve, m = 0; m < VEND / 4; ++m, 1+=4)
         if(1 \ge VEND) 1-=VEND;
        *(long far *)(c_blue + 1) = 0;
         *(long far *)(c red + 1) = 0;
         *(long far *)(c green + 1) = 0;
         *(long far *)(c itsty + 1) = 0;
               /* sign=1ならば不要部分の消去をする */
   else {
      for (m = 0, p = qve; m < 20; ++m, p+=4) {
  for (1 = 0, q = p; 1 < SCLDOT; ++1, q+=80) {
            *(long far *)(c_blue + q) = 0;
```

までんできたは、マナーと、花は 第一の一十一

```
*(long far *)(c red + q) = 0;
     *(long far *)(c green + q) = 0;
           *(long far *)(c_itsty + q) = 0;
  while((inportb(0xa0) & 0x4) == 0);
  outportb(0x00a2, 0x70); /* スクロールコマンドの送出 */
  sclad1 -= SCLPTC;
  if(sclad1 <= GSTADR) {
    sclad1 = GENADR;
    sclno1 = 0;
    sclno2 = STLINE;
     adl.dataw = GSTADR;
     ad2.dataw = GENADR - SCLPTC;
     nol.dataw = ENLINE;
     no2.dataw = SCLD16;
  else {
     sclno1 += SCLD16;
     sclno2 -= SCLD16;
     adl.dataw = sclad1;
     ad2.dataw = GSTADR;
     nol.dataw = sclnol;
     no2.dataw = sclno2;
 nol.datab.high |= gdchz;
  no2.datab.high |= gdchz;
  while((inportb(0xa0) & 0x20) != 0);
  while((inportb(0xa0) & 0x20) == 0);
  outportb(0xa0, adl.datab.low); /* GDCへ各パラメータを送る
  outportb(0xa0, adl.datab.high);
  outportb(0xa0, no1.datab.low);
  outportb(0xa0, no1.datab.high);
 outportb(0xa0, ad2.datab.low);
  outportb(0xa0, ad2.datab.high);
  outportb(0xa0, no2.datab.low);
outportb(0xa0, no2.datab.high);
  if(sign == 0) { /* sign=0なら全画面を砂漠のデータで埋める */
 for(1 = gvs; 1 < VEND; 1+=2) {
  *(int far *)(c_blue + 1) = 0;
  *(int far *)(c_red + 1) = 0xaaaa;
 *(int far *)(c green + 1) = 0xaaaa;
        *(int far *)(c_itsty + 1) = 0xaaaa;
```

```
else { /* sign=1ならばマップデータを表示する */
for (map x = 0; map x < 20; ++map x, qvs+=4, qve+=4) {
   n = map data[map y][map x];
for(1 = 0, m = map ptn y, p = gvs, q = gve;
   char low 1 < SCLDOT; ++1, ++m, p += 80, q += 80 ) {
    *(long far *)(c blue + p) = pattern[n].dtb[m];
          *(long far *)(c red + p) = pattern[n].dtr[m];
""" bara *(long far *)(c_green + p) = pattern[n].dtg[m];
   *(long far *)(c_itsty + p) = pattern[n].dti[m];
   strugt EXTE databi
 map ptn y -= SCLDOT; AGATED = wsjsb.lbs
   if (map ptn y < 0) {
   map ptn y = PTY - SCLDOT; 3MIJMS = wajsb.lon
   ++map y;
   if (map_data[map_y][0] == 0 || map y >= 240) map y = 0;
 enemy sign = gvs - 80 * 2;
   for(n = 0; n < 20; ++n) {
   enemy_table1[n] = map data[map y][n];
   001012
   Bap pth y,
   else enemy sign = 0;
```

さて、LIST4-4.Cでは、スクロールと共にマップの表示処理も同時に行っています。マップの表示開始位置を示しているのが関数の引数 gvsで、表示終了位置を示しているのが gve です。すなわち、gvs の位置でスクロールした分だけマップを表示し、gve の位置で不要部分の消去をするわけです。

引数の sign は画面の初期状態を表示するためのサインを意味しています。sign=0で初期画面の表示及び使用変数の初期化作業をしています。では、この関数を実際に動かしてみましょう。main()関数とプロジェクトファイルは次のようになります。なお、プログラムの先頭では使用するパターンを格納してあるファイルとマップデータを格納してあるファイルをロードする構造としました。

LIST4-5.C /* スクロールのテスト */ #include <graphics.h> / 11/1/ / / 2 / / Albag Lalab gam #include <conio.h> #include <stdio.h> #include <bios98.h> #include <dos.h> #define PTY 32 80*PTY #define NPAT #define MAKE ESC 0 #define SCLDOT 2 #define PTY0 #define VSTADR #define MAKE RETURN 0x1c extern int pattern in (unsigned int, int); extern int pattern_out(unsigned int, int); extern int map_data_load(void); extern void scroll(int, int, int); extern char gdchz; int i, j, k, 1, m, n, py0, pye, keydat; KEY_INFO keyinf; if(inportb(0x31) & 0x80) gdchz = 0; else gdchz = 0x40; grphinit(); pointer_set(); printf("\nn 4-> + - + - + "); if(gload() == 0) { for $(k = m = n = 0, l = mwindow; k < 4; ++k, l+=NPAT) {$ for(i = 0, j = 1; i < 11; ++i, j+=4) { n = pattern_in(m++, j);

```
if(n != 0) break;
   if (n != 0) break; when you want a grant - n a x
   } /n = map data(map y)[map x];
  map_data_load(); = map_pts_y, p = gvs, gr. offderp> abuleal4
  clrscr(); < sclbor; ++1, ++m, p += 80, q += 80
  keydat = 0; ong tar +) (g god + p) = pattern | b of the
  pv0 = 80 * VSTADR;
   pye = 400 * 80 - 80 * SCLDOT;
   scroll(80 * VSTADR, py0, 0);
   while (keydat == 0) {
    keydat = bios98key(&keyinf);
   scroll(py0, pye, 1);
     py0 -= 80 * SCLDOT;
     if(py0 < 0) py0 = 80 * 400 - 80 * SCLDOT;
     pye -= 80*SCLDOT;
     if(pye < 0) pye = 80 * 400 - 80 * SCLDOT;
   extern int pattern in (unsigned int, int); x = 0 + x = 0
else {
  printf("\formall n press any key"); (blov)bsol_sjsb_qsm ini misjxs
  extern void scroll(int, int, int);
closegraph();
```

```
LIST4-5.PRJ

list2-0
list2-1
list3-0
list4-0
list4-2
list4-4
list4-5
```

■ ヒーロー登場

いかがでしたか、実際にデザインしたマップがスクロールしていく過程を見るのは一種感動するものがあるのではないでしょうか。さて、いよいよ作り出した新しい世界の中を動き回るヒーローの登場と願いましょう。

あるパターンをグラフィック画面上に表示する場合、パターンの余白から見えるはずの背景をいかに表示するかという重ね合わせ処理が問題となります。

これを完全に処理するためには、どの部分がパターンで、どこからが余白であるのかを表す専用のデータが必要となります。データ構造としてはパターンがある所は0、無い所は1として背景との論理演算 AND を実行することで背景をパターンの形でくり抜くという操作をするわけです。後は、くり抜いた背景の所に論理演算の OR でパターンを表示すればよいのです。

実は、このように言葉で説明するのは簡単なのですが、これをプログラムするとなると、かなり複雑になってしまいます。そこで本書では、この重ね合わせの問題を解決するために、砂漠(口絵6)以外には移動できないという制限を付けることにします。すなわち、パターンの余白の部分にあらかじめ背景の砂漠のパターンで埋めておくわけです。このような制限を付けることによってプログラムは飛躍的に簡単になります。しかも、パターンの移動後に残る残骸を元の背景に復元する処理も単純に砂漠を描けばよいわけですから、さらに処理が簡単になるわけです。もっとも、マップを描く時には必ず砂漠が存在しないと移動できないということになりますから注意してください。では、LIST4-6.Cへと進んでください。

			LIST4-6.C	
#define	PTY	32		tSU = zib
#define	NPAT	80*PTY		keep px = px;
#define	MAKE_RIGHT	0x48		

```
#define MAKE LEFT 0x46
#define BREAK RIGHT 0xc8
#define BREAK LEFT 0xc6
#define XMAX
                                         76
#define XMIN
                                いかがでしたか、実際にデザインしたマッナが見から
#define ON
                                         1
#define OFF 0 TAXABLE TO A AMOUNT A THE MARKET TO A LIFE
#define CONTINUE 0
#define UE 1 A TOTAL AND THE WAY OF THE AND TH
#define MIGI
                                         。見えるはぎの背景をいかに表示するかぞいう電影
                                         3
#define HIDARI
#define VEND 400*80
extern char far *c blue;
extern char far *c_red;
extern char far *c green;
extern char far *c_itsty;
extern int keydat;
long dtb[PTY];
        long dtr[PTY];
    long dtg[PTY]; DOXAMBLOS TRUST REDICTOR
        long dti[PTY]; 。 tまいましてしない推薦になか、よるなるなす。
 pattern[50];
int check(unsigned int, int, int);
int px = 38;
/* 車の表示及び移動処理 */
int car_disp(int py, int keep_py)
   static int
                dir_m = OFF,
                 dir h = OFF,
                pn = 0; y = 0;
        unsigned int j, k;
        int dir, keep px, r;
        dir = UE;
        keep px = px;
```

```
switch (kevdat) {
     case BREAK RIGHT: > # Tyg + TAGM + mg great * 100 * 5000
     dir m = OFF; (GMEV == 1 (GMEV =< 1)31
     if (dir h == ON) keydat = MAKE LEFT;
       else break; sssassk0 = ( + ber b) (* rat pro!) *
  case MAKE LEFT: sassassio = (f + neero o) (* 187 prof)*
       dir = HIDARI; (0 = (f + vjsti o) (* asi pool) *
       dir h = ON;
 for (k = 0, j = px + 3 + py; k < PTY; j+=80, ++k)
  if(j >= VEND) j -= VEND;
          *(c blue + j) = 0;
   *(c red + j) = 0xaa;
  *(c green + j) = 0xaa;
(x)dsbq(0)ano *(c itsty + j) = 0; (* rsl paol)*
*(long far *) (d red + }) == ([0].dum[k]) *
(A) A3b (O if (px < XMIN) { ydedl o) (* 381 pnol) *
dir = UE;
px = XMIN;
mwiitob(c)cr) (
 1 4% break; t : xrq > x : yq + xq = t .0 = x) xox
  case BREAK LEFT: (CMAY -- (CMAY -< ())
(x)daba(& dir h = OFF; + suid s) (* ast paol) *
if (dir m == ON) keydat = MAKE RIGHT;
(Alpab | else break; - + mass place ret profits
case MAKE RIGHT:
       dir = MIGI;
       dir m = ON;
     for (k = 0, j = px + py; k < PTY; j+=80, ++k) {
   if(j \ge VEND) j = VEND;
          *(c_blue + j) = 0; (avav + c_b)
(diddb (Slaze * (c red + j) = 0xaa; sal prol) *
*(c green + j) = 0xaa;
(x) pub (S) are * (c_itsty + j) = 0; (* ral pro!) *
*YIGHG Tarysq (60888ty weg) is patterd[2].dtl[k]:
       ++px;
       if (px > XMAX) {
          dir = UE; (c green a pry) sasard : Musteb
          px = XMAX;
       break:
    default :
       dir m = OFF;
       dir h = OFF;
      break;
```

```
for (k = 0, j = keep px + NPAT + py; k < 8; j+=80, ++k) {
if(i >= VEND) i -= VEND;
*(long far *)(c blue + j) = 0; ( (0) == 4 xib) 11
*(long far *)(c red + j) = 0xaaaaaaaa; and sale
#define *(long far *)(c_green + j) = 0xaaaaaaaa; %1 %3%M eess
if (check (px + py, keep_px + keep_py, dir) == CONTINUE) {
switch(pn) { (GMSV =- } (GMSV =< {)}}
define with case 0:
for (k = 0, j = px + py; k < PTY; j+=80, ++k)
               if(j >= VEND) j -= VEND;
               *(long far *)(c_blue + j) = pattern[0].dtb[k];
                *(long far *)(c red + j) = pattern[0].dtr[k];
   *(long far *)(c_green + j) = pattern[0].dtg[k];
     *(long far *)(c_itsty + j) = pattern[0].dti[k];
  break;
  for (k = 0, j = px + py; k < PTY; j+=80, ++k) {
               if(j >= VEND) j -= VEND; 17931 XA388 8880
   *(long far *)(c blue + j) = pattern[1].dtb[k];
   *(long far *)(c red + j) = pattern[1].dtr[k];
   *(long far *)(c green + j) = pattern[1].dtg[k];
   *(long far *)(c itsty + j) = pattern[1].dti[k];
| nettern (501: }
           break;
  chark() case 2: wif all a had (yg + xg = 1 0 = x) 101
   for (k = 0, j = px + py; k < PTY; j+=80, ++k) {
                if(j >= VEND) j -= VEND;
                *(long far *)(c blue + j) = pattern[2].dtb[k];
  *(long far *)(c_red + j) = pattern[2].dtr[k];
      *(long far *)(c_green + j) = pattern[2].dtg[k];
                *(long far *)(c itsty + j) = pattern[2].dti[k];
            }
      break;
      default : break;
      ++pn;
      if(pn > 2) pn = 0;
      r = 0;
   else {
    px = keep_px;
```

```
1383: r = 1; A C いるのがなる 私 多はあっかり x x がかかかり 行するり と たた C
dir_m = OFF; (figra + bergs) (* 181 page) * q
     dir h = OFF; (1000 4 neers 5) (4 1st phot) 4 - p
 keydat= 1;
  return(r); LIST4-7.CH+-#-K
1- 1- 1-
/* 移動方向が移動可能かを判断する */
int check (unsigned int pxy, int keep pxy, int dir)
  unsigned char a, b, c; Sx0 a (0ax0) disagni) sildw
  unsigned int pxy2; (05x0 a (0ax0)dayogab)) alldw
  int i, j, k, l, r; wag good = ( .0 = x) and
  long o, p, g;
  pxy2 = pxy; 1980 = 00004 . ber b) (* 183 pnoi)*
  r = CONTINUE; sq = 1000 meorp o) (* rst pool) *
 | switch (dir) | { sq = (f + ysast b) (* xai pool) *
  case MIGI:
  pxv += 3;
  case HIDARI:
   a = *(c_blue + pxy); / (yxg good = 1,0 = 4)x0
        b = *(c red + pxy); GMEV =- } (GMEV =< }) %
   fraudo c = *(c green + pxy); * said o)(* rst paol)*
        if(a == 0 && b == 0xaa && c == 0xaa) {
       pxy += 80 * 16;
   if (pxy >= VEND) pxy -= VEND;
           a = *(c blue + pxy);
  b = *(c red + pxy);
  c = *(c_green + pxy);
   if(a == 0 && b == 0xaa && c == 0xaa) {
              pxy += 80*16;
              if (pxy > VEND) pxy -= VEND;
              a = *(c blue + pxy);
               b = *(c red + pxy);
            c = *(c_green + pxy);
               if(a == 0 && b == 0xaa && c == 0xaa);
              else r = CRASH;
           トに数値を使って指定する為の工夫なのですが、こと
            else r = CRASH;
      else r = CRASH;
case UE:
```

```
o = *(long far *)(c blue + pxy2);
p = *(long far *)(c red + pxy2);
   q = *(long far *)(c green + pxy2);
   if (o == 0 && p == 0xaaaaaaaa && q == 0xaaaaaaaa);
   else r = CRASH;
   * ci break; *) (a creac + 1) = (mananana) (1) nuisi
   default : break;
if (r == CRASH) {
   for(i = 0; i < 3; ++i) ( geek fal yxg fal bengianu) keeds fal
      for(1 = 3; 1 \le 5; ++1) {
         printf("Ya");
         while ((inportb(0xa0) & 0x20) != 0); and beneland
         while ((inportb(0xa0) & 0x20) == 0);
            for (k = 0, j = keep pxy; k < PTY; j+=80, ++k) {
            if(j >= VEND) j -= VEND;
            *(long far *)(c_blue + j) = pattern[l].dtb[k];
           *(long far *)(c red + j) = pattern[l].dtr[k];
         *(long far *)(c green + j) = pattern[1].dtg[k];
        *(long far *)(c itsty + j) = pattern[l].dti[k];
         Fortk = 0, 5 = pm + py; k < PTY; 5:4800H eamp (
  for(k = 0, j = keep_pxy; k < PTY; j+=80, ++k) {
      if(j >= VEND) j -= VEND; (yxanta dos oktomedi) deg(k)
      *(long far *)(c blue + j) = 0;
      *(long far *)(c red + j) = 0xaaaaaaaa;
      *(long far *)(c green + j) = 0xaaaaaaaa;
      *(long far *)(c itsty + j) = 0;
return(r);
```

さて、ここでの処理のポイントは、ヒーローとなる車をデザインしたパターン番号の0~2を順番に表示する処理の所で switch~case 文を使用していることです。これは、構造体を参照する場合のインデックスをダイレクトに数値を使って指定する為の工夫なのですが、この方が、変数による場合よりも若干スピードが早いということです。

また、移動可能か否かのチェックをG.V.RAMから直接データを読んで判断しています。砂漠のデータは、B面が0で、R面とG面のデータが

0xaaaa······と並んでいる構造となりますから、車の進行方向と左右で G. V.RAM の数値をチェックするわけです。なお、左右でチェックする場合には、パターンの上中下の3箇所でチェックをします。では実際に動かしてみましょう。なお、LIST4-7.C はキーボード設定用となりますから注意してください。

```
LIST4-7.C。 $P$$P$$P$(0.4.2) 0 1871 - 第一并 *\
#include <dos.h>
#include <bios98.h>
#define STATUS PORT
                 0x43
#define COMMAND PORT
                0x43
#define DATA PORT
                 0x41
#define EOI
                 /* キーボード割り込みを復元する */10 = s.f.obg (68xg 0.
#define COMMAND
                 0x16
#define EOIDATA
                 0x20
#define ERRCODE
                 0x38
int keydat,
   random= 3751,
   seed2 = 1357;
void frandom(void);
/* キーボード割り込み関数の定義 */
void interrupt keyset (void)
  int x; Data load();
  frandom();
   x = inportb(STATUS PORT) & ERRCODE;
   if(x == 0) {
      outportb (COMMAND PORT, COMMAND);
      keydat = inportb(DATA PORT);
   outportb(EOI, EOIDATA);
/* 疑似乱数用関数 */
void frandom(void) 38, (3nl ,int bengismu) ni nrejing
    pattern out unsigned int, inthissy * 08 = 0vq
   seed1 = random; 4 40 - 40 + screen (blov)bsolp
   random <<= 2;
```

```
random += seed1;
  ++random;
random = random & 0xf;
 random +=2;
ましょう。なお、LIST4-7.Cはキーボード設定用となりますから注意
void interrupt (*keepvector9)();
void interrupt keyset (void);
/* キーボード割り込みの初期設定 */
void initkey (void)
  keepvector9 = getvect(9);
  setvect(9, keyset);
/* キーボード割り込みを復元する */
             "(long far ") (o blue 48.590 ...
void termkey (void)
  KEY_INFO keyinf; KEY_INFO keyinf;
  setvect(9, keepvector9);
  keyinf.cmmd = 3;
  bios98key(&keyinf);
```

#include <graphics.h> #include <conio.h>

```
#include <stdio.h>
#include <dos.h>
               32
#define PTY
#define NPAT
                80*PTY
#define MAKE ESC
               0 .
#define SCLDOT 2 (TWOS ATAG) ASSOCIATED TRANSPORT
#define PTY0
                42*80*8
#define VSTADR
               16
#define MAKE RETURN 0x1c
extern int keydat, px;
extern int pattern in (unsigned int, int);
extern int pattern out (unsigned int, int);
extern int gload(void);
extern void initkev (void);
```

```
extern void termkey (void);
extern int car disp(int, int);
extern void scroll(int, int, int);
extern int map data load(void);
extern char gdchz;
int mwindow = 36;
void main()
              int i, j, k, l, m, n, py, py0, pye, r;
        unsigned int kyori;
               if(inportb(0x31) & 0x80) gdchz = 0;
          else gdchz = 0x40;
             clrscr();
             printf("\xn\" \quad \n' \quad \n' - \quad \mathread \mathread \n' \quad \n' 
              for (k = m = n = 0, 1 = mwindow; k < 4; ++k, 1 += NPAT) {
                       for(i = 0, j = 1; i < 11; ++i, j += 4) {
                           n = pattern_in(m++, j);
                            if(n != 0) break;
                            STATE T
                            if (n != 0) break; property and a property of the property of 
                            Har far *c blue;
                             map_data_load();
                          clrscr();
                          initkey();
                            gotoxy(1, 1);
                          printf("RALLY 98");
                          keydat = 1;
                          py0 = 80 * VSTADR;
                              while (keydat != MAKE_ESC) {
                        scroll(80 * VSTADR, py0, 0);
                       i = PTY0;
              py = PTY0;
       px = 38;
        py0 = 80 * VSTADR;
      pye = 400 * 80 - 80 * SCLDOT;
      keydat = 0x70;
```

```
kyori = 0x0;
   extern int oar disp(int, int))
   gotoxy(68, 1); (381 381 381) Horse blow mratke
   printf("距離"); (blow bsol sisb qsm jnl hasike
        while (keydat != MAKE ESC && r == 0) {
           gotoxy(73, 1);
world intermed printf("%u m
                       ", kyori);
           ++kvori;
r = car_disp(py, i);
void initiay (wo while (keydat == 0x70);
           scroll(py0, pye, 1);
  keepvectors i = py; t(%h) syq .0vq .vq .n .m .1 .x .f .1 sml
  py -= 80 * SCLDOT; throys and bengiana
           if (py < 0) py = 80 * 400 - 80 * SCLDOT;
py0 -= 80 * SCLDOT; dobo (08x0 a (18x0) dirocal) %1
if(py0 < 0) py0 = 80 * 400 - 80 * SCLDOT;
           pye -= 80 * SCLDOT;
  if(pye < 0) pye = 80 * 400 - 80 * SCLDOT;
   setweet ik, keenvector9);
  Reputh Commed = 3:
   termkey();
  for(k = m = c = 0, 1 = mwindow; k < 4; +fk, 1 += NFAT) |
  printf("Yn press any key"); and anothing - a
     getch();
closegraph();
```

```
LIST4-8.PRJ

list2-0
list2-1
list3-0
list4-0
list4-2
list4-4
list4-6
list4-7
list4-7
```

■敵の出現

敵、すなわち、進行を邪魔する存在ですが、本書では砂漠の中から突然、回転しながら現れる三角形の板としました。そうです、アニメーションのサンプルで使ったあのパターンです。アニメーションといってもパターン番号を順番に変化させるという簡単なものです。ここで特に難しいのは出現させるタイミングでしょう。進行を妨げるのですから、移動不可能な砂漠以外の所に出現したのではあまりにも芸がないからです。次のプログラムでは、疑似乱数を使ってマップデータから砂漠のパターンをサーチして、敵を起動するタイミングとしてあります。では、一気に最後のプロジェクトファイルである LIST4-10.PRJ まで進んでください。

```
LIST4-9.C
                      /* パターンを構成するライン数 */
#define PTY
#define SCLDOT 2
                      /* スクロールドット数 */
#define VEND
                      /* G.V.RAMのENDアドレス */
           400*80
#define SABAKU 6
                      /* 砂漠パターン番号 */
                      /* 敵パターンアニメーションスタート番号 */
#define STAPTN 7
#define ENDPTN 18
                      /* 敵パターンアニメーションエンド番号
                      /* B面へのポインタ */
extern char far *c blue;
                      /* R面へのポインタ */
extern char far *c red;
                      /* G面へのポインタ */
extern char far *c green;
extern char far *c itsty;
                      /* I面へのポインタ */
                      /* 敵x座標決定用 */
extern int enemy table1[20];
                      /* 敵出力サイン */
extern int enemy sign;
                      /* 乱数格納用 */
extern int random;
extern int px;
                      /* パターンx座標 */
/* パターン格納用構造体 */
extern struct {
  long dtb[PTY];
long dtr[PTY];
long dtg[PTY];
long dti[PTY];
```

```
/* 敵の状態管理用構造体の完義 */
struct {
  int inf;
  int x;
  int dv:
int ptn:
        ンプルで使ったあのパターンです。アニメーシ旨の対しい
void frandom(void);
/* 敵の表示用関数の定義 */
void enemy disp(int enemy set)
は外の所に出現したのではあまりにあ締結ないから かた、次のプログ
  int i, j, k, l, m, n;
  if (enemy set == 1) {
for(i = 0; i < 4; ++i) {
       /* 敵の状態別処理 */
       switch(enemy[i].inf) {
          case 0:
           /* 敵表示開始処理 */
       if (enemy sign != 0) {
       frandom();
        for(j = 0, k = random; <math>j < 20; ++j) {
    printf("We goest verse ++k; o o *\ 00*00$ ONTV ani's
            if (k \ge 20) k = 0;
  j = enemy_table1[k];
  OLOSO GRANDO CARROLL STATE OF SABAKU) {
             enemy[i].inf = 1;
                    k = k + k + k + k;
                            = k;
neerp o* rel rado
                    enemy[i].x
                    enemy[i].y = 16;
                    enemy[i].dy = enemy_sign+k;
              enemy[i].ptn = STAPTN * 4;
             enemy_sign
              break;
             }
           break;
          case 1:
            /* 敵Y座標更新 */
            enemy[i].y += SCLDOT; [YTT] | SD prod
            if (enemy[i].y > 400-10*PTY) enemy[i].inf = 2;
```

```
printf("MA break; # 3 %) 8 *08 * SA
  case 2:
  /* 敵パターン更新処理 */
  n = enemv[i].ptn;
     n >>= 2;
     if(n >= ENDPTN) enemy[i].inf = 0;
     for(j = 0, m = enemy[i].dy, k = PTY-1;
     py * PTYR j < PTY; ++j, --k) { (blow) bsolp jat greixe
            *(long far *)(c_blue + m)
                 = pattern[n].dtb[k];
             *(long far *)(c_red + m)
     = pattern[n].dtr[k];
       *(long far *)(c green + m)
                  = pattern[n].dtg[k];
           *(long far *)(c itsty + m)
     = pattern[n].dti[k]; max blov dxedke
     m -= 80;
     while (keyds if (m < 0) m += VEND; 0) ( tadobp tado miejke
       gotoky (73. 1):
       ++ enemy[i].ptn;
       break;
       default : break;
else {
  /* 敵情報テーブル初期化 */
  for(i = 0; i < 4; ++i) enemy[i].inf = 0;
enemy_sign = 0;
       py0 -- 80 + SCIDOT)
```

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <dos.h>
#include <dos.h>
#define PTY 32
#define NPAT 80*PTY
#define MAKE_ESC 0
#define SCLDOT 2
```

```
#define PTY0 42*80*8
#define VSTADR 16
#define MAKE_RETURN 0x1c
extern int keydat, px;
extern int pattern in (unsigned int, int);
extern int pattern_out(unsigned int, int);
extern int gload(void); (%-- , ++ (YT9 > +
extern void initkey(void);
extern void termkey(void);
extern int car_disp(int, int);
extern void scroll(int, int, int);
extern int map_data_load(void);
extern void enemy_disp(int);
extern void frandom(void);
extern char gdchz;
int mwindow = 36;
void main()
   int i, j, k, l, m, n, py, py0, pye, r;
   unsigned int kyori;
   if(inportb(0x31) & 0x80) gdchz = 0;
                      qdchz = 0x40;
   else
   grphinit();
   pointer set();
   clrscr();
   printf("\n' \f-\) \f' -\f \U-\'");
   if(gload() == 0) {
      for (k = m = n = 0, l = mwindow; k < 4; ++k, l += NPAT) {
         for (i = 0, j = 1; i < 11; ++i, j+=4) {
            n = pattern_in(m++, j); <d.soldqsrp> ebulonit
            if(n != 0) break;
         if(n != 0) break;
      map data load();
      clrscr(); YTG*08
      initkev();
      gotoxy(1, 1); seement a sub-likery content (shows well-set
```

```
printf("RALLY 98");
   kevdat = 1;
   py0 = 80 * VSTADR;
   while (keydat != MAKE ESC) {
       enemy_disp(0);
       scroll(80 * VSTADR, py0, 0);
       i = PTY0;
       py = PTY0;
       px = 38;
       py0 = 80 * VSTADR;
       pye = 400 * 80 - 80 * SCLDOT;
       keydat = 0x70;
       kvori = 0x0;
       r = 0;
       gotoxy(68, 1);
       printf("距離");
       while (keydat != MAKE ESC &&
           gotoxy(73, 1);
           printf("%u m
                          ", kyori);
           ++kyori;
           r = car_disp(py, i);
           while (keydat == 0x70) frandom();
           enemy_disp(1);
           scroll(py0, pye, 1);
           i = py;
           py -= 80 * SCLDOT;
           if(py < 0) py = 80 * 400 -
           py0 -= 80 * SCLDOT;
           if(py0 < 0) py0 = 80 * 400 - 80 * SCLDOT;
           pye -= 80 * SCLDOT;
           if(pye < 0) pye = 80
                              * 400
   termkey();
else {
   printf("Yn press any key");
closegraph();
```

	42*40*8LIS	ST4-10.PRJ Y J J A S *) 13 ml rq
list2-0		py0 = 80 * VSTADR;
list2-1		
list3-0		while(keydat != MAKK_ESC) ;
list4-0		
list4-2		soroll(80 * VSTADEL pyec, es
list4-4		
list4-6		py = PTY0;
list4-7		28 * XQ
1ie+4-9		
list4-10		
extern int	cer dispinat, inth	. pve = 400 * 80 - 80 * SCLDC

実行してみると、たった一種類の敵ですが今までにない雰囲気となったはずです。ここでは一種類の敵だけとしましたが、さらに多くの敵を出現させ、攻撃パターンに工夫を凝らしたりすれば、このゲームもより面白いものになると思います。

なお、機種によっては実行スピードが遅く感じられる場合があるかと思いますが、その時には、MS-DOSのコマンドラインから直接起動してみてください。

さて、サンプルプログラムということで、背景描写や敵のパターンも含めて33種類のパターンだけでゲームを作ってみました。市販のゲームと比べると今一歩の感は否めないようですが、それもそのはずです。市販のゲームでは、数百ものパターンを用意するのが普通です。しかも、プログラム、パターンデザイン、グラフィックデザイン、シナリオ、音楽というように、それぞれの分野のプロが専門分野を担当するというのが一般的ですから、無理もないことなのです。

とはいいながら、アイデアしだいでは世界中に受け入れられるゲームが 生まれる可能性がありますから、後は読者の創造性にまかせることとして 本書でのゲーム作りの幕としましょう。

map_data_lead();
clrsor();
initkey();

第全次重ね合わせ処理や。左右のドット単位スクロール処理、多重スクロー

C言語によるゲーム作りはいかがでしたか、このようにグラフィックス 処理も遊び感覚でできるゲームから入ると楽しく進められたのではないで しょうか。プログラミング作業も視覚で確かめながら進めていくことがで き、しかも、ゲームは様々なテクニックが要求されるために、プログラム テクニックの修練の場としては最適といえるのではないでしょうか。

グラフィックス処理には様々な分野がありますが、基本はG.V.RAMへのポインタをどのように扱うかに尽きると思います。CAD、CAMに代表される CG の世界も、ファミコンに代表される TV ゲームの世界も、すべて G.V.RAM をどのように扱うのかという問題へと行き着いてしまうのです。

PC-9801シリーズは、CAD、CAM や TV ゲーム用の専用機ではありませんが、グラフィックス処理専用のハードウェアである GDC (Graphic Display Controller) や GRCG (Graphic Charger)、EGC (Enhanced Graphic Charger)などが登載されているために、かなりの処理が可能になります。これらのハードウェアのコントロールも、本書程度の G.V.RAM の処理を把握していないと、やはり困難であるといえます。

ところで、この GDC に関しては、それだけで一冊の本ができるぐらいの 内容がありますので、本書では GDC の持っている機能の中でもスクロー ル機能だけに絞り、他の機能を解説することはしませんでしたが、その他 様々な機能を応用すれば、たとえば直線描画機能を光線銃の軌跡に利用す るなど、さらにおもしろいものができると思います。

GRCGやEGCに関しては、登載機種が限られているために、サンプルプログラムでの使用を避けました。しかし、特にEGCには、シフト機能を有した4画面同時転送機能や、ビット演算機能、さらにEGCとGDCとを組み合わせて使うなど、ゲーム制作にはうってつけの機能が豊富にあります。登載されている機種であれば、これらの機能を使わないのは、まさに「宝

の持ち腐れ」といったところではないでしょうか。たとえば、パターンの完全な重ね合わせ処理や、左右のドット単位スクロール処理、多重スクロールなどアイデアしだいで応用範囲はかなり広いものがありますから、ぜひ、EGC の活用も研究してみて下さい。

また、TURBO CではTURBO アセンブラと組み合わせることによって、Cのプログラム中にインラインアセンブラとしてアセンブリ言語のプログラムを書くことができます。もし、TURBO アセンブラがなくても、MASM で開発したマシン語とのリンクが簡単にできるために、EGC などの機能が登載されていない機種を使用している方や、汎用性の観点からPC-9801シリーズ固有の GDC、GRCG、EGC というハードウェアは使いたくないならば、C言語とマシン語のリンクという方法を検討してみるのもよいでしょう。もちろんこれらの登載機種であっても、たとえばパターン表示関数などのような、特にスピードを要求される一部の処理をマシン語に置き換えるというのは、度々用いられている手法です。

本書をステップに GDC や EGC を駆使した本格的なグラフィックス処理にチャレンジしてみてください。もしも、マシン語やゲームに興味がある方であれば、『はじめてのマシン語』(啓学出版)、『マシン語ゲームプログラミング』(アスキー)『マシン語ゲームグラフィックス』(小学館)などにも取り組んでみてください。また、最後になりましたが、本書および続編に対する御意見、御希望などありましたら聞かせていただければ幸いです。

様々な機能を応用すれば、たとえば直線描画機能を光線鉄の軌跡だ利用学 答会と、¹をなたなを包含記書の数号を名を展びます。ここれないいおよ

GRCG や EGC に関じてほ、登載機種が限られているために、サンツルブ カグラムでの使用を繰りました。しぬけ、場合と対策など。

た4曲面同時転送機能や、ビット演算機能、さらに EGC と GDC とを組み といけて描された。 ゲース組作にける。てつけの場所が無常によります。

登載されている機能であれば、これらの機能を使わないのは、まさに「宝

索引品。例是以外不多的证券以

- **あ** アスキーコード 7 アニメーション処理 126
- (2) 色の成分のチェック 41色の混ぜ合わせ 29インタープリタ 1一の実行過程 2インラインアセンプラ 171
- え エンハンストグラフィック チャジャー 16
- お オープン 107 オフセット 10 ---アドレス 10,21 オブジェクトファイル 2
- か カーソル 46,69,84
 一形状データ 48
 一の表示 56
 拡大表示 64
 重ね合わせ処理 155
 型変換 130
- き キー情報 76 キーボード 76 一割り込み 76,113 キャスト 130
- く クローズ 107 グラフィックスシステム 16,18 グラフィックスドライバ 16,19 グラフィックチャージャー 16 グラフィックビデオラム 11

- 構造体 127コピー 95コンパイラ 2――の実行過程 2コンパイル 2
- シフト演算子 39,48実行プログラム 2条件式の評価 42初期化 19初期化作業 18
- **す** スクロール 145 ──コマンド 147 ──コマンドのパラメータ 148 スモールモデル 17
- せ セーブ 107,113 セグメント 10 ---アドレス 10,21
- そ ソースプログラム 1
- **て** ディップスイッチ 149 ディレクトリ内のファイル表示 115

デバイスドライバ 53

- と 動作クロック 148 ドット座標 35 ドット表示 56
- に ニブル 6

- たところではないでしょ ハードウェアスクロール 145 メモリモデル 10.17.74 汎用 BOX ルーチン 61
- 一一アドレス 36 ーイメージ 11,126 操作 103
- ふ ファィル管理 107 ファィルの捜索 115 ファイルの入出力管理 115 ファィルポインタ 107 ファィル名 113 ファィル名の入力 115 ファンクションキー 90 ファンクション機能 90 フィールド 24 プレーン 11.21 プロジェクトファイル
- ベクタテーブル 74 変数の可視性
- マウス 53

---ドライバ 53

ードライバの組み込み手順

割り込み マシン語コード 1

マップエディタ 133

マップデータ 133,165

め メモリアドレス 36 メモリ管理 10

- も モニタ 9
- ライブラリ 16
- 107,113 3
- ワイルドカード 113 割り込み関数 74 割り込み処理 52,53,69 割り込みテーブル 70 割り込みベクタ 69 割り込みベクタテーブル 52 割り込みマスクレジスタ 70 割り込みを許可 59 割り込み禁止 59

10進数と4桁の2進数の対応 4 1桁の2進数の足し算 4 2進数 4 2進数の単位 6 2次元配列 133 600×200ドットモード 13 600×400ドットモード 12

英字

B BASIC 9 BGI 16 BGI.ARC 17 binary number 4 binary-digit 6 bit 6

GRCG 16,171 Blue(青) 21 Borland Graphics Interface 16 Green(緑) 21 BOX ルーチン H Hexadecimal 9 I if 文 42 cast 130 initgraph() 18,19 Central Processing Unit 10 Intensity(輝度) 21 interrrupt 修飾子 70.74 アースながまなり1面 21 いちでいれき detectgraph() 16.18 EGC 16.171 M MK FP() 22 Enhanced Graphic Charger 171 N NEC版マウスドライバ 53 exit(1) 19 nibble 6 far タイプのポインタ 21 P PC98.BGI 16 fclose() 107 PC98EGC.BGI 16 fgetc() 107 PC98GRCG.BGI 16 FIFO バッファ 146 putimage() 126 findfirst() 115 Red(赤) 21 findnext() 115 R First In First Out 146 R面 21 fopen() 107 T TURBO アセンブラ fputc() 107 UNPACK.EXE 17 H G G.V.RAM 11,20,21,22 unsigned 40,48 GDC 145,171 V V-SYNC割り込み GDC のステータスフラグ 147 void 型 getimage() 126 Graphic Charger 171 Graphic Display Controler X XOR(排他的論理和) 46 145,171 XOR の真理値 46 GRAPHICS.H 17 GRAPHICS.LIB 17

Y

Y軸 35

C

E

F

graphresult() 19

[プログラムソースのディスクサービスのお知らせ]

本書に掲載したプログラムのソースコードとパターンデータの入ったディスクを販売いたします。ご希望の方は巻末に添付してある振り替え用紙に住所、氏名、電話番号、希望のディスクメディアをご記入の上、お申し込み下さい。

価格:3000円 (送料込み)

- ・本ディスクにはTURBO CおよびBG1ファイル、MS-DOSシステムは含まれていません。
- ・本ディスク中のプログラムをそのまま、あるいは改変して 使用した結果として生じた損害について、著作者および出版 者はいっさいの責任を負いません。
 - ・プログラムから Copyright(c) Manabu Aoyama の表記を削除しないで下さい。

[著者紹介] **** *** 青山 学

1958年 東京生まれ。

青山学院大学理工学部卒業。コンピュータエンジニアを経て、 現在はゲームデザイナー。

大型コンピュータからパソコンまで、言語、機種にこだわらないのを信条としている。著書として、『PC-9801シリーズはじめてのマシン語』『8086マシン語秘伝の書』(以上啓学出版)、『PC-9801シリーズマシン語ゲームプログラミング』(アスキー)、『マシン語ゲームグラフィックス』(小学館)などがある。

趣味は、スキー、テニスなど。

PC-9801+TURBO C グラフィックスに強くなる本 ゲーム編 © 青山学 1991

1991年9月30日 第1刷発行

著者青山 学

発行所 啓学出版株式会社 代表者 三井数美

郵便番号 101

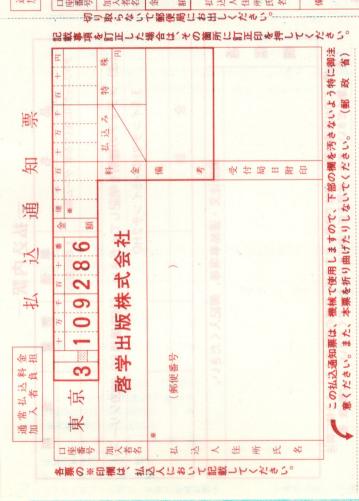
東京都千代田区神田神保町1-46 電 話 東京03(3233)3795[販売部] 東京03(3233)3731[編集部]

振 替 東京 3-109286

印刷/昭和工業写真印刷所製本/徳 住 製 本 所

ISBN4-7665-0900-5

本書の定価はカバーに表示してあります。



9 申 学出版株式会社 00 4 + 票 S Į. Þ 0 + 払込 + F H + 受付局日附印 通常払込料金加 入者負担 3 + 凼 宁 垂 丰 老 加入者名 色 払込人住所氏名 口座審导

このたびは、小社の通信販売を ご利用いただき、有難うござい ます。

○この払込票は、本領収書となります。 ります。 ○送金手数料は、弊社負担です。 ○この払込票は、書籍のご注文 にもご利用いただけます。

一払込内訳一

西 数 量	마	3.5インチ2HD・5インチ2HD	(注) メディアの種別を明記してください。(対応機種PC-9800シリーズ)	10 O	グラフィックスに強くなる本 ゲーム編 3,000(税込	品 名 単
		design of the same a state of the same	い。(対応機種PC-980	0 1 0 2 4	兑込)	価数量

藏 前 通 追加注文・連絡事項等、御記入ください。

意ください。また、本票を折り曲げたりしないでください。

(郵 政 省)

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注

內 啓学出版株式会社

